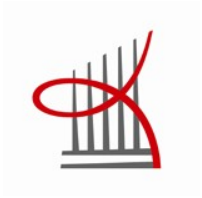


TIETOTURVALLINEN WWW-OHJELMOINTI

Jaakko Waltzer

Opinnäytetyö
Toukokuu 2012
Tietojenkäsittelyn koulutusohjelma

TAMPEREEN AMMATTIKORKEAKOULU

Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma

WALTZER, JAAKKO: Tietoturvallinen WWW-ohjelmointi

Opinnäytetyö 57 sivua
Toukokuu 2012

Opinnäytetyö käsittelee tietoturvan perusteita ja tietoturvallista WWW-ohjelmointia. Työn kohderyhmä on aloittelevat WWW-ohjelmoijat, hiukan ohjelmointia osaavat sekä tietoturvasta yleisellä tasolla kiinnostuneet henkilöt. Opinnäytetyössä kerrataan tietoturvan periaatteet lyhyesti ja käydään läpi tavallisimmat verkkouhkat, joista pääpaino on injektiohyökkäyksissä (XSS-hyökkäys, SQL-injektio).

Työn tavoitteena oli kartoittaa keskeiset verkkouhat ja -hyökkäykset ja etsiä ratkaisut niiden torjumiseksi. Lähestymistapa verkkohyökkäyksiin on käytännönläheinen. Siksi työssä selvitetään kokeellisesti hyökkäyksen toteuttaminen. Valtaosa hyökkäyksistä simuloitiin suojaamattomassa testiympäristössä. Suojaamattoman testiympäristön tarkoitus oli osoittaa konkreettisesti hyökkäyksen toteutus ja vaikutus WWW-palveluun. Kokeellisten hyökkäyksien lisäksi raportissa käydään läpi menetelmiä, joilla hyökkäyksiä voidaan ehkäistä. Menetelmiä täydennetään tarkastelemalla hakemistorakenteen suunnittelua, tiedostojen ja hakemistojen oikeuksia ja suojausta sekä PHP:n konfigurointia tietoturvan kannalta kriittisiltä alueilta. Työ sisältää runsaasti esimerkkejä sekä hyökkäyksistä että toimintamalleja niiden torjunnasta.

Opinnäytetyön tuloksena syntyi tietoturvan perusopas. Oppaan keskeinen tarkoitus on auttaa kohderyhmää sekä tiedostamaan verkkohyökkäysten vaarallisuus että oppia yksinkertaiset menetelmät niiden ehkäisyyn. Opas ei ole kaikenkattava, mutta toimii hyvänä perustyökalupakkina, jonka avulla kohderyhmä voi rakentaa turvallisempia WWW-sovelluksia. Lisäksi oppaan tarkoitus on auttaa lukijaa ymmärtämään tietoturvan merkitys WWW-ohjelmoinnissa.

Asiasanat: tietoturva, PHP, MySQL, WWW-ohjelmointi

ABSTRACT

Tampere University of Applied Sciences
Degree Programme in Business Information Systems

WALTZER, JAAKKO: Secure Web Programming

Bachelosr's thesis 57 pages
May 2012

This thesis deals with basic information security and secure web programming. The main target groups for this thesis are web programmers and people who are interested in information security. The thesis summarises information security principles and finds out general network threats (like XSS-attacks and SQL-injection).

The goal of the thesis was to clarify general network threats and find out methods for preventing them. The network attacks are approached from a practical viewpoint. This means that the attacks were simulated in a vulnerable test web service. The main reason using a vulnerable testing environment was to indicate how easily an attacker can make an attack and how the attacks affect web services. In addition, the thesis looks at methods to prevent these attacks. These methods are also complemented by examining the design of the directories and rights for those directories and their protection. The thesis also looks at PHP configuration in critical information security areas. This thesis contains many examples of attacks and effective ways to prevent them.

As a result of this thesis a basic guide to information security was created. The fundamental objective of this thesis is to help its target audience to acknowledge the danger of web attacks and learn a few simple ways for preventing them. The guide does not cover every possible web attack, but it offers good tools for its target audience to build safer web applications. In addition, this guide is meant to give a wider understanding of information security in web programming.

Keywords: information security, PHP, Mysql, web programming

SISÄLLYS

TERMIT

AJAX
Selainpuolen tekniikka jonka avulla käyttäjä voi tehdä muutoksia WWW-sivulle, siten ettei koko sivustoa tarvitse ladata muutoksen jälkeen uudelleen.

Apache
Suosittu avoimeen lähdekoodiin perustuva WWW-palvelin.

DOM
W3C:n kehittämä ohjelmointirajapinta, jonka avulla voidaan käsitellä HTML- ja XML-dokumentteja.

HTML
Merkkauskieli jolla kuvataan WWW-sivun rakennetta ja esitystapaa.

JavaScript
Netscape Communications Corporationin kehittämä selainpohjainen skriptikieli, jolla voidaan sisällyttää WWW-sivuille dynaamista toiminnallisuutta.

MVC
Arkkitehtuuri jossa sovellus jaetaan 3 osaan model, view ja controller. Model piilottaa tietovaraston toiminnan muilta osilta, view näyttää tiedot ja controller käsittelee sovelluksen pyynnöt.

MySQL
Relaatiotietokantajärjestelmä joka on saatavissa sekä vapaalla GNU GPL-lisenssillä että kaupallisella lisenssillä.

PDO
PHP:n versiosta 5.1 alkaen lisätty luokkajärjestelmä, jolla voidaan hallita useita erilaisia tietokantoja.

PHP
The PHP Groupin 1995 kehittämä skriptikieli, jolla voidaan ohjelmoida dynaamisia WWW-sovelluksia palvelinympäristössä.

Olio-ohjelmointi
Ohjelmointitekniikka jossa ohjelma muodostuu joukosta keskenään keskustelevia olioita.

VBScript
Microsoftin kehittämä skriptikieli, jolla voidaan ohjelmoida palvelimella olevia ASP-sivuja ja selainpohjaista toiminnallisuutta WWW-sivuille.

1 JOHDANTO

Dynaamisten verkkopalveluiden käyttö tietoverkoissa on saavuttanut suunnattomat mittasuhteet. Esim. Facebookin maailmanlaajuiseksi käyttäjämääräksi arvioitiin alkuvuonna 2012 noin 800 miljoonaa (Wikipedia 2012). Facebookin omien tilastojen mukaan käyttäjiä oli peräti 845 miljoonaa joulukuun lopussa 2011(Facebook 2012).

Myös muita yhteisöpalveluja käyttää suunnaton määrä käyttäjiä, esim. Twitteria arvioitiin käyttävän maaliskuussa 2011 yli 200 miljoonaa käyttäjää (Wikipedia 2011a). Yhteisöpalveluiden lisäksi julkiset palvelut, yritykset, yhteisöt sekä yksityiset tarjoavat lukuisia dynaamisia verkkopalveluita.

Palveluiden siirtyminen verkkoon aiheuttaa valtavien tietomäärien käsittelyä, jolloin tietoturvan tulisi olla ensisijainen lähtökohta sivustoa tai verkkopalvelua suunniteltaessa. Varsinkaan yhteisöpalveluiden tietoturvaan ei ole kiinnitetty riittävää huomiota. Esim. Dicitals Societyn 2.4.2011 tekemän testin perusteella yhteisöpalveluiden tietoturva ei vastaa niille asetettuja vaatimuksia. Asteikolla (A-F) testattiin 11 verkkopalvelua, joista ainoastaan Gmail, Wordpress (SSL) saivat arvosanan A. Suosituimmat yhteisöpalvelut Facebook ja Twitter saivat arvosanan F. (Ou 2011)

Julkisilla verkkopalveluilla tietoturva on huomattavasti parempi johtuen osaltaan valtionvarainministeriön julkaisemasta ohjeistuksesta Julkisten verkkopalveluiden laatukriteerit. Tästä huolimatta tietoturva ei toteudu joka osa-alueella. Oikeusministeriön mukaan vuonna 2008 järjestettiin kunnallisvaaleissa sähköinen äänestys Karkkilassa, Kauniaisissa sekä Vihdissä. Sähköisiä ääniä annettiin yhteensä 12 234 kappaletta, joista 223 oli keskeytynyt. Tapauksessa ilmenee, ettei tiedon saatavuutta pystytty varmistamaan riittävästi. (Oikeusministeriö 2008.)

Yrityksillä tietoturva on vaihtelevalla tasolla riippuen toimialasta. Pankkien ja vakuutusyhtiöiden tietoturva on kohtuullisella tasolla, mutta jatkuvasti raportoidaan hyökkäyksistä ja salasanojen kalasteluista em. yrityksiin. Myös suuret teknologiayhtiöt ovat hakkerien suosiossa. Suurin ongelmana ovat ns. tavoitteelliset hyökkäykset, joissa hyökkääjä hakee taloudellista tai muuta hyötyä yrityksestä. Pienten yritysten verkkopal-

veluihin kohdistuvista hyökkäyksistä raportoidaan harvemmin, joten näiden tietoturvataso on usein arvoitus.

Opinnäytetyön aihe on tietoturvallinen WWW-ohjelmointi. Opinnäytetyö tarkoituksena on kartoittaa tietoturvaan liittyvät keskeiset ongelmat ja uhkat, sekä löytää niihin käytännönläheiset ratkaisut. Opinnäytetyössä pääpaino on WWW-sovelluksen kehittäminen tietoturvallisesti, mutta myös tietoturvan perusteita käsitellään suppeasti.

Aihe opinnäytetyöhöni syntyi työharjoittelupaikassani, jossa tehtäväni oli toteuttaa pienehkö verkkojulkaisu työnantajan ohjeistuksen mukaan. Harjoittelupaikkani oli Oy Terästö Ltd, joka on pieni julkaisutoimintaa harjoittava yritys. Projektia tehdessäni jouduin kiinnittämään erityistä huomiota tietoturvaan liittyvistä asioista ja huomasin, että useat verkkojulkaisuoppaat käsittelevät tietoturvaa varsin pintapuolisesti. Tämä ja jatkuvat raportoinnit verkkohyökkäyksistä, vahvistivat opinnäytetyön aiheen hyödyllisyyden.

Opinnäytetyö ei ole suoraan sidoksissa mihinkään verkkosovellukseen, mutta opinnäytetyössä sovelletaan työharjoittelussa opittuja ratkaisuja, joiden pohjalta simuloidaan testisovelluksella erilaisia hyökkäyksiä ja menetelmiä niiden torjumiseksi. Hyökkäyksien pääpaino on kahdessa hyökkäystyypissä Cross Site Scripting (XSS) ja SQL-injektio. Näiden lisäksi tarkastellaan pintapuolisesti muita tietoturvauhkia.

Testien ja lähdemateriaalien perusteella luodaan tietoturvan perusopas, jolla aloitteleva ohjelmoija, ohjelmoinnista ja/tai tietoturvasta kiinnostunut henkilö oppii ymmärtämään tietoturvan merkityksen, sekä menetelmät yleisimpien hyökkäysten torjumiseksi. Oppaassa keskitytään PHP- ja MySQL-ohjelmointi kieliin Apache-ympäristössä, mutta perusajatus on tietoturvallinen WWW-kehitys yleisesti.

Työssä on käytetty sekä kirjallisia että verkkolähteitä. Vanhin kirjallinen teos on julkaistu jo vuonna 2002. Teos käsittelee tietoturvan osa-alueita, joidenka määritelmät eivät ole muuttuneet. PHP- ja MySQL-ohjelmointia käsittelevät kirjalliset lähteet ovat julkaistu vuosina 2005, 2010 ja 2011, joista vanhinta (suomennettu teos) olen käyttänyt lähinnä aihepiiriin tutustumiseen.

Vanhin kirjallinen ohjelmointia koskeva teos ei sellaisenaan ole hyvä lähde, mutta sisältöanalyysia käyttämällä sitä voitiin soveltaa joissakin tilanteissa. Muut kirjalliset lähteet ovat melko uusia, mutta kirjallinen aineisto on kuitenkin vanhempaa kuin verkkoaineisto. Edellisestä johtuen kaikkien lähteiden tutkimisessa on käytetty tekstivertailua, jolla on pyritty varmistamaan sisällön oikeellisuus.

Verkkolähteet tukevat kirjallisia lähteitä, koska niistä löytyy uusin ohjelmointia koskeva tieto. Lisäksi runsaat testit tukevat kirjallisia lähteitä ja varmistavat, että tieto on ajankohtaista.

Opinnäytetyössä ei käsitellä varsinaisia tietoturvaan liittyviä tutkimuksia kovinkaan syvällisesti, vaan tukeudutaan tilastotietoihin ja raportteihin erilaisista hyökkäyksistä ja muista uhkista esim. CERTI-FI:n verkkosivusto, joista saa ajankohtaista tietoa verkkohyökkäyksistä.

Oppaan teoriaosuudessa on kolme keskeistä teemaa:

- Tietoturva yleisesti (käydään lyhyesti läpi tietoturvan peruskäsitteitä).
- Tietoturva verkossa (tutkitaan erilaisia hyökkäyksiä ja niiden vaikutuksia WWW-palveluihin).
- Turvallinen WWW-ohjelmointi (käydään läpi tietoturvallisen WWW-sovelluksen rakentamisen peruselementit).

2 TIETOTURVA YLEISESTI

Tietotekniikka koskettaa nyky-yhteiskunnassa meitä jokaista. Julkisen sektorin tietojärjestelmät varastoivat monia erilaisia tietoja jokaisesta kansalaisesta. Myös lukuisat yritykset, yhteisöt sekä yksityishenkilöt tuottavat verkkopalveluita, joihin vaaditaan käyttäjiltä erilaisia tunnistetietoja. Laaja tietojen varastointi asettaa suuria haasteita tietoturvalle, joihin vastaaminen vaatii tietoturvan perusteiden ymmärtämistä. Tässä luvussa käydään lyhyesti läpi tietoturvan peruskäsitteitä.

Tietoturvan periaatteet

Tietoturvan määritelmä ja tavoitteet eivät ole itsestäänselvyys, koska aihe on laaja ja sitä lähestytään usein organisaatioiden näkökulmasta. Järvisen (2002) mukaan tietoturva on jaettu karkeasti kolmeen osa-alueeseen (CIA), tiedon luottamuksellisuus, tiedon eheys ja tiedon saatavuus. Osa-alueita on täydennetty kolmella välttämättömällä perusperiaatteella, todentaminen, pääsynvalvonta ja kiistämättömyys, joita ilman kolme osa-aluetta ei voi toteutua. (Järvinen 2002, 22–28.) Useat tutkijat ja tietoturvan ammattilaiset pitävät määritelmää kuitenkin liian epämääräisenä tai tarkoitukseen sopimattomana johtuen sen laajuudesta.

Luottamuksellisuus

luottamuksellisuus tarkoittaa sitä, ettei kukaan pääse käsiksi tietoon, johon hänellä ei ole oikeutta (Järvinen 2002, 22). Luottamuksellisuuteen liittyy olennaisina tekijöinä käsitteet todennus ja salaus, joilla pyritään varmistamaan, ettei ulkopuolinen pääse käsiksi oikeudettomasti tietoon. Hyvänä esimerkkinä luottamuksellisuudesta voidaan käyttää pankin tietojärjestelmiä, jossa suojattuun ympäristöön päästään useiden erilaisten tunnusten yhdistelmällä, joihin kuuluu mm. kertakäyttöinen tunnusluku, joka on osoittautunut melko luotettavaksi menetelmäksi suurta luottamuksellisuutta vaativissa palveluissa.

Eheys

Eheys tarkoittaa sitä, ettei kukaan pysty oikeudettomasti, tai vahingossa muuttamaan tietoa. Esimerkkinä eheyden rikkoutumisesta voi olla esim. tiedoston poistaminen vahingossa tai tallennuslaitteen vahingoittuminen. Pahimmassa tapauksessa eheyden rikkoutuminen voi tapahtua ulkopuolisen tahon toimesta esim. tiedon sisällön muuttamisella, joka saattaa olla haitallisempaa kuin tiedon katoaminen. Tiedon eheyden turvaa-

misen liittyy olennaisesti erilaiset tarkistusrutiinit, tarkistussummat lokitiedostot ym. (Järvinen 2002, 22–23.)

Saatavuus

Saatavuus (käytettävyys) tarkoittaa sitä, että tieto tai palvelu on aina saatavissa silloin, kun tietoa on oikeus käyttää. Saatavuuteen voi vaikuttaa negatiivisesti esim. palvelunestohyökkäykset tai järjestelmissä ilmenevät puutteet. (Järvinen 2002, 24.) Palvelun saatavuudessa ilmennevät ongelmat saavat usein paljon julkisuutta palveluissa, joita käyttää suuri määrä käyttäjiä. Esimerkkinä voidaan mainita VR:n uusien lippujärjestelmien käyttöönotto, joka jouduttiin sulkemaan järjestelmän ongelmien vuoksi (Vaalisto 2011).

Todentaminen

Todentamista tapahtuu jatkuvasti. Korttiostot, korttinostot ja verkkopankinkäyttö ym. ovat palveluja, joissa joudutaan aina todentamaan käyttäjä. Järvisen mukaan todentamisella varmistetaan palvelun tai sovelluksen käyttäjän aitous riippumatta siitä, onko kysymyksessä ihminen, sovellus, WWW-sivu tai ohjelmakoodi. Olennaista todentamisella on, että käyttäjällä on sekä tietyt oikeudet palveluun tai sovellukseen että tiettyyn oikeuteen kuuluvat identifiointitunnukset. Käyttäjän todentaminen perustuu yleensä käyttäjätunnusten ja salasanojen yhdistelmään kun kysymyksessä on luonnollinen henkilö. (Järvinen 2002, 24–27.)

Pääsynvalvonta

Todennettujen käyttäjien pääsystä järjestelmään tai palveluun huolehtii pääsynvalvonta. Pääsynvalvonta on sovellustason menetelmä, johon kuuluu mm. käyttäjäseuranta, jonka lokitiedostoista voidaan havaita mahdolliset väärinkäytökset tai niiden yritykset. (Järvinen 2002, 27.)

Kiistämättömyys

Kiistämättömyys liittyy sähköiseen kaupantekoon, jossa tilaukset, toimitukset, tilisiirrot ym. muut sopimukset on pystyttävä todistamaan lain edellyttämällä tavalla. Kiistämättömyys vaatii edellisten periaatteiden toteutumista, jotta mahdollinen liiketapahtuma voidaan todeta. (Järvinen 2002, 22 – 28.)

Tietoturvallisuuden käsitteiden ymmärtäminen ei sinänsä riitä turvaamaan tietoturvallista verkkopalvelukehitystä, vaan lisäksi käytetään kansainvälisiä ja/tai kansallisia tietoturva standardeja ja yrityksien omaa tietoturva politiikkaa, jossa määritellään, miten organisaatiotasolla tietoturva kokonaisuudessaan turvataan. Viestintäviraston (2012) mukaan organisaatioiden ja käyttäjien toimenpiteet on jaettu lähtökohtaisesti kahdeksaan tietoturvan osa-alueeseen:

- hallinnollinen turvallisuus
- henkilöstöturvallisuus
- fyysinen turvallisuus
- tietoliikenneturvallisuus
- laitteistoturvallisuus
- ohjelmistoturvallisuus
- tietoaineistoturvallisuus
- käyttöturvallisuus.

3 TIETOTURVA VERKOSSA

Tässä luvussa käydään läpi lyhyesti joitakin yleisiä verkon tietoturvauhkia sekä laajemmin verkkosovelluksissa olevia yleisiä haavoittuvuuksia.

3.1 Yleisiä tietoturvauhkia

3.1.1 Phishing

Phishing on huijaus, jossa pyritään keräämään uhrin henkilökohtaisia tietoja, kuten pankkitunnuksia ja salasanoja tai luottokorttitietoja ym. Uhrille voidaan lähettää sähköpostiviesti, jossa uhrilta kysellään henkilökohtaisia tietoja hyvämaineisen yrityksen nimiin tai uhri ohjataan väärennettyyn verkkopalveluun. Väärennetty verkkopalvelu on usein visuaalisesti lähes identtinen oikean verkkopalvelun kanssa tai muutoin luottamustaherättävän näköinen. Kirjautuessaan väärään verkkopalveluun uhri luovuttaa huomaamattaan käyttäjätunnuksensa ja salasanansa väärälle palvelulle. (Microsoft 2011.)

Huijauksen tekee usein mahdolliseksi uhrin tietämättömyys tai huolimattomuus sekä hyvin väärennetty sivusto, joka lisää huijauksen uskottavuutta. Huijauksissa voidaan käyttää XSS-hyökkäysmenetelmää, jolloin esimerkiksi sähköpostilla lähetetyn linkin osoitteen alkuosa on yrityksen oikea osoite, ja loppuosa scripti, joka ohjaa käyttäjän huijaussivustolle. Huijauksista varoitetaan julkisesti, kun huijausyritys havaitaan, mutta silti huijaukset saattavat toisinaan onnistua.

Suomessa verkkopankeissa käytetään tunnuksien ja salasanojen lisäksi kertakäyttöistä avainta, joka parantaa turvallisuutta, mutta uhrin hyväuskoisuuteen perustuvassa huijauksessa siitä ei ole aina apua. Viimeisin julkisuuteen tullut onnistunut huijaus tapahtui alkuvuonna 2010, jolloin Nordean 15:tä asiakkaalta onnistuttiin viemään yhteensä 50 000 € (Lehto 2010).

3.1.2 Käyttäjän huolimattomuus

Käyttäjän oma huolimattomuus verkossa saattaa aiheuttaa henkilökohtaisten tietojen joutumista vääriin käsiin. Tottumaton verkkopankin käyttäjä saattaa jättää istunnon auki poistuessaan julkiselta koneelta. Joku voi käyttää toistuvasti samoja helposti johdettavia salasanoja, jotka saatetaan murtaa pelkästään tuntemalla henkilön taustaa, jonka henkilö on kertonut esim. omalla kotisivullaan. Joku taas voi kirjautua jokaiseen puutaheinää palveluun ja mahdollisesti luovuttaa palveluun tietoja, joita ei saisi luovuttaa ulkopuolisille.

Salasanojen heikkoudesta hyvä esimerkki on Bogdan Calin artikkeli 6.10.2009, jonka mukaan yleisin Hotmail-palvelun salasana oli 123456. Vertailussa oli 10000 salasanaa joista 123456 esiintyi peräti 64 kertaa (Calin 2009).

Käyttäjän toiminta jää pitkälti käyttäjän omalle vastuulle, mutta verkkosovelluksen rakentaja voi joissakin tapauksissa parantaa käyttäjän tietoturvaa:

- istunto katkaistaan, jos toimintaa ei ole tapahtunut tiettynä aikana
- käyttäjä voidaan pakottaa käyttämään riittävän pitkiä ja monimutkaisia salasanoja
- salasana vanhenee määräajassa ja käyttäjän on vaihdettava se
- oheistus on riittävän selkeä ja kattava (kohderyhmän segmentointi) ym.

3.2 Yleisimmät haavoittuvuudet

Verkkohyökkäys perustuu usein WWW-sovelluksessa olevaan haavoittuvuuteen, johon ohjelmoija ei ole kiinnittänyt riittävästi huomiota. Asiaan voi vaikuttaa ohjelmoijan kokemattomuus tai yleinen tietämättömyys todellisista tietoturvaukista. Monissa verkkopalveluissa on havaittu vakavia puutteita, vaikka palveluja ylläpitää useita ammattilaisia. Lisäksi monissa julkaisujärjestelmistä havaitaan haavoittuvuuksia, joihin verkkojulkaisija ei aina osaa kiinnittää huomiota.

Haavoittuvuuksien ja uhkien tunnistaminen auttaa sekä ohjelmoijaa että julkaisujärjestelmän käyttäjää, suunnittelemaan toimintatapoja ja menetelmiä, joilla voidaan ennaltaehkäistä useita verkossa olevia uhkia. Tässä luvussa käydään läpi yleisimpiä haavoittuvuuksia sekä muita verkkouhkia.

Opinnäytetyössä käytetyt esimerkit ovat pelkistettyjä ja toteutettu suojaamattomassa ympäristössä. Tämän tarkoitus on havainnollistaa hyökkäyksien toteutus mahdollisimman selkeästi. Todellisuudessa hyökkääjä saattaa käyttää helppokäyttöistä, internetistä ladattavaa ohjelmaa, jolla hyökkäyksen voi toteuttaa tuntematta lainkaan ohjelmointitekniikoita.

3.2.1 HTTP Heder Injection

HTTP Header Injectionissa hyökkäys kohdistuu HTTP-otsikokenttiin tai evästeeseen, jossa on käyttäjän antamaa tietoa. Otsikkokenttä paloitellaan siten, että yhdestä HTTP-pyynnöstä muodostuu kaksi vastausta, joista toisen voi päättää hyökkääjä. HTTP header injectonilla voidaan aiheuttaa mm. XSS-haavoittuvuuksia, joista puhutaan luvussa 3.2.2. (Siu 2007.)

3.2.2 Cross-Site Scripting (XSS)

Cross-Site Scripting on yksinkertainen hyökkäysmenetelmä, jossa voidaan syöttää vahingollista koodia suojaamattomalle sivustolle. Esimerkiksi Java Script- tai VB-scriptikielellä hyökkääjä voi toteuttaa sivustolla ei toivottua toiminnallisuutta. Pelkällä HTML-merkkauskielelläkin voidaan aiheuttaa vakavia tietoturva uhkia. Hyökkäyksien tarkoitus voi olla ottaa sivusto hallintaan tai ohjata käyttäjä toiselle sivustolle. Toiminta saattaa olla uteliaan henkilön kokeilua, mutta usein hyökkäykset ovat tavoitteellisia, joilla pyritään keräämään käyttäjien tietoja esim. käyttäjätunnuksia ja salasanoja ym. XSS-hyökkäykset voidaan jakaa karkeasti kolmeen tyyppiin: pysyviin, ei pysyviin ja DOM-pohjaisiin (Wikipedia 2011b).

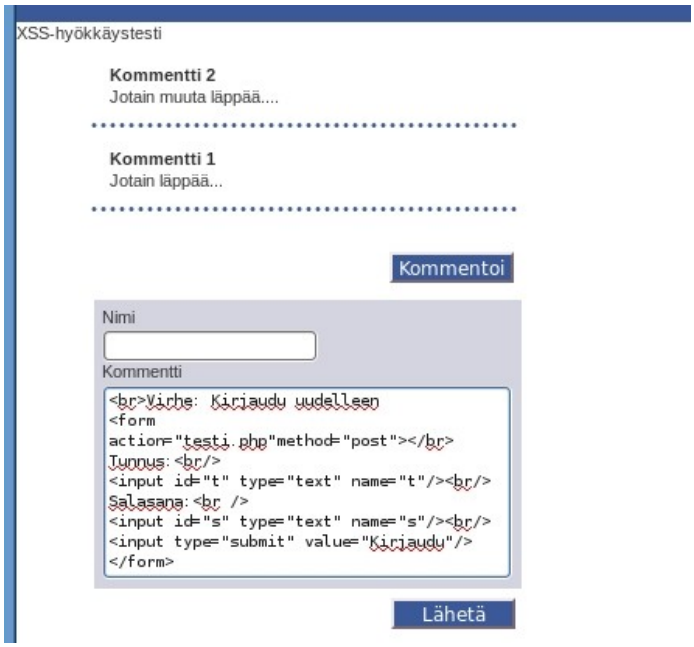
Pysyvässä XSS-hyökkäyksessä hyökkääjä syöttää vahingollista koodia, joka jää pysyvästi sivuston verkkopalvelimelle. Hyökkääjä voi syöttää koodia mm. lomakkeelta tietokantaan, jonka jälkeen koodi ajetaan sivua päivitettäessä. (OWASP 2011a.) Koodi toimii siten, että suoritettaessa se käyttäytyy kuin se kuuluisi sivuston koodiin. Kuvan 1 esimerkissä käydään läpi yksinkertainen pysyvä XSS-hyökkäys, jossa hyökkääjä pyrkii keräämään verkkopalvelun käyttäjien käyttäjätunnuksia ja salasanoja. Hyökkäys on toteutettu yksinkertaisella HTML-lomakkeella, jolla kysytään käyttäjän käyttäjätunnusta ja salasanaa.

Esimerkissä käyttäjillä on mahdollisuus kommentoida WWW-palvelun sisältöä. Kommenttipainike tuo esiin kommentti lomakkeen, jolla kommentin voi jättää. Kommentti lisätään WWW-palvelun tietokantaan. Sivu päivittyy kommentin jättämisen yhteydessä tai sivulle tultaessa.

Lähtötilanne: Sivulle on jätetty kommentteja, jotka näytetään käyttäjälle niiden jättämisen jälkeen (kuva 1).

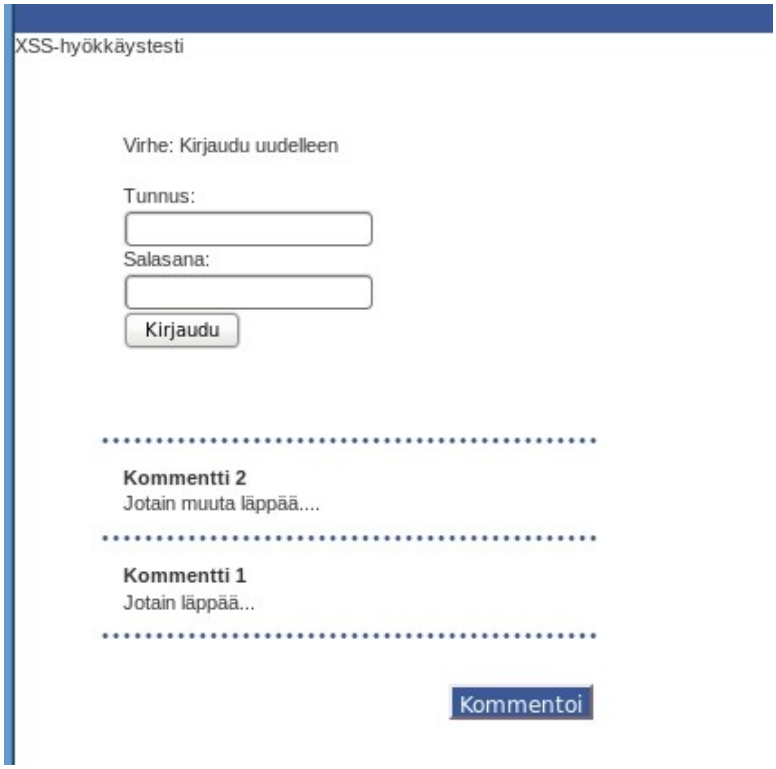
KUVA 1. WWW-palvelun kommenttinäkymä

Hyökkääjä haluaa kerätä palvelun käyttäjien tunnuksia ja salasanoja hyödyntämällä sivustolla olevaa kommenttitoimintoa. Kommenttikenttään lisätään HTML-lomake (kuva 2).



KUVA 2. Kommenttilomake

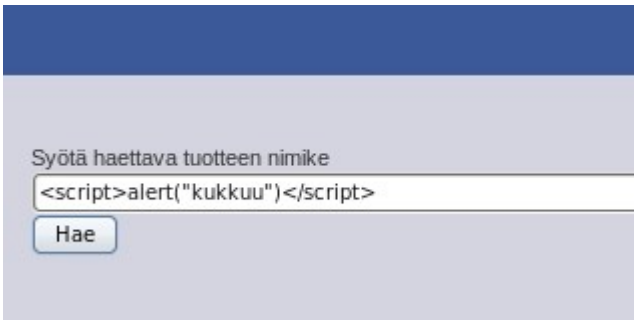
Sivun päivittymisen jälkeen käyttäjät näkevät lomakkeen, jossa ilmoitetaan virheestä ja pyydetään uudelleen kirjautumista (kuva 3). Mikäli käyttäjä täyttää esille tulleen lomakkeen, lähetetään käyttäjän tunnus ja salasana hyökkääjän haluamaan paikkaan.



KUVA 3. Kommenttinäkymässä näkyvä huijauslomake

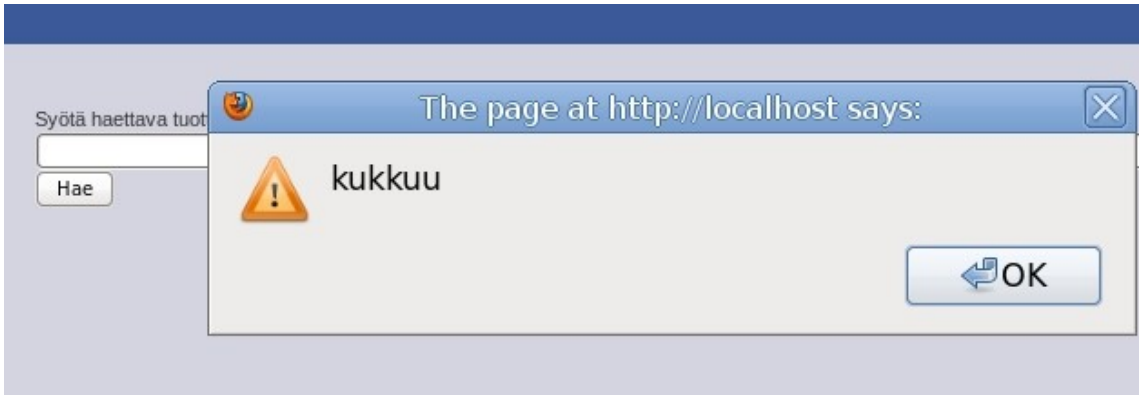
Ei-pysyvä XSS-hyökkäys on menetelmä, joka voidaan toteuttaa sivuston ulkopuolisten linkkien avulla. Tyypillinen ei-pysyvä XSS-hyökkäys on scriptin sisältävä sähköpostilinkki, jossa WWW-osoitteen perään on lisätty skripti, joka ohjaa käyttäjän luotetun sivuston kautta väärennetylle sivustolle. (OWASP 2011b.) Linkki voi sisältää myös skriptin, joka ohjaa oikealle sivulle, mutta luo sivustolle väärän kirjautumislomakkeen, jolla voidaan kalastaa käyttäjätunnuksia ja salasanoja. Menetelmää on käytetty lukuisissa huijausyrityksissä.

Yksinkertaisessa XSS-hyökkäyksessä voidaan esim. hakukenttään kirjoittaa pieni koodinpätkä, joka suoritetaan hakutoimintoa painaessa (kuva 4).



KUVA 4. Yksinkertainen hyökkäysskripti

Kuvan 4 skripti tulostaa ilmoituksen (kuva 5).



KUVA 5. Hyökkäysskriptin tuottama ilmoitus

Hyökkäys saattaa vaikuttaa harmittomalta, mutta kertoo hyökkääjälle, ettei sivulla ole suodatusta ja hyökkäys on toteutettavissa. Seuraava vaihe voi olla esim. se, että muutetaan skripti lomakkeeksi ja lähetetään osoitelinkkinä uhrille. Uhri klikkaa linkkiä ja saa-

puu oikealle sivulle. Sivu on oikea, mutta sivulle tulostuu huijauslomake, joka kysyy esim. käyttäjätunnusta ja salasanaa.

DOM-pohjainen XSS-hyökkäys on selainpohjainen hyökkäys, jossa käyttäjien syötettä käytetään osana sivun sisältöä. DOM-pohjainen hyökkäys on saman kaltainen kun ei pysyvä XSS-hyökkäys, mutta käyttää hyödykseen esim. JavaScript *document.referer*,- *window.name*-, ja *location*-ominaisuuksia. Hyökkäys toimii siten, että esim. JavaScript-koodi saa syötteen käyttäjältä. Koodi välittää parametrina annetun syötteen, jolloin hyökkääjä voi syöttää vahingollista koodia esim. osoiteriviltä. Tuloksena parametri saa arvokseen hyökkääjän koodin. (OWASP 2011b.)

3.2.3 Cross Site Request Forgery (CSRF)

CSRF on hyökkäysmenetelmä, joka on XSS-hyökkäyksen vastakohta. CSRF-hyökkäyksessä väärennetään HTTP-pyyntöjä käyttämällä tavallisesti img-tageja. Hyökkäys toimii siten, että käyttäjä on kirjautunut jollekin sivulle ja käy samaan aikaan toisella sivulla, johon on sisällytetty hyökkäystagi. Käyttäjä klikkaa hyökkäyslinkkiä ladatakseen esimerkiksi sivulla olevan kuvan, jolloin hyökkääjän haluama toimenpide suoritetaan. Käyttäjän käyttämällä palvelimella näyttää siltä kuin hyökkäyskoodi tulisi sivulta, johon käyttäjä on kirjautunut. (MacIntyre, Danchilla & Gogala 2011, 251.)

3.2.4 SQL-injektio

Myös SQL-injektio on yleinen ja melko yksinkertainen menetelmä, jonka avulla hyökkääjä pystyy hyödyntämään WWW-palvelussa tai sovelluksessa olevan haavoittuvuuden syöttämällä komentoja tietokantaan, joita sinne ei saa syöttää. Useimmiten kysymyksessä on vahingollinen koodi, jonka tarkoitus on hyödyttää hyökkääjää. SQL-injektion avulla hyökkääjä voi suorittaa erilaisia operaatioita tietokantaan esim. poistaa, muuttaa tai varastaa tietokannan tietoja.

Pahimmassa tapauksessa hyökkääjä voi luoda tietokantaan root-tasoisen käyttäjän, jolla tietokannan hallitseminen onnistuu täydellisesti. SQL-injektion tekee mahdolliseksi useimmiten lomakekenttien ja muun syöttötiedon puutteellinen validointi.

SQL-injektion toteutus

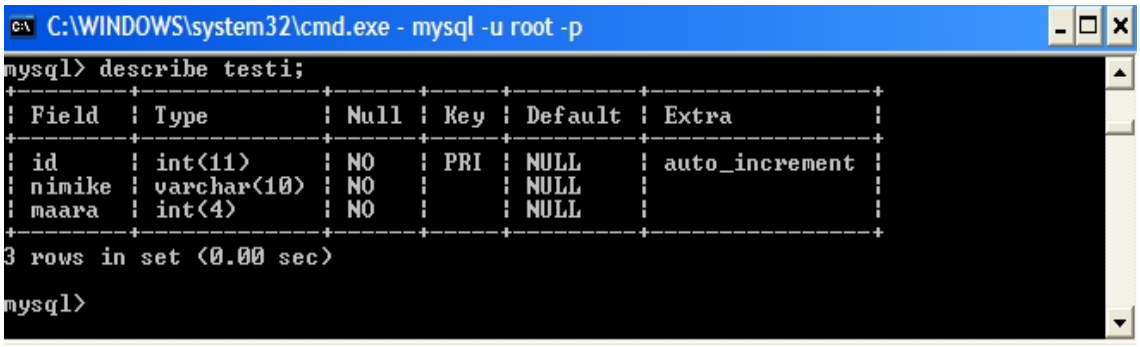
Tyypillisen tietokantahyökkäyksen toteuttaminen onnistuu varsin helposti suojaamattomalla sivustolla. Hyökkääjä saattaa aluksi pyrkiä selvittämään tietokannan rakennetta esim. tietokannan ja taulujen nimet. Tämän jälkeen hyökkääjä määrittelee, mitä tietoja tietokannasta voidaan hyödyntään, ja miten hyökkäys toteutetaan. Hyökkäyksiä voi olla käytännössä yhtä paljon kuin on erilaisia tietokantakyselyjä.

Hyökkäys toteutetaan muokkaamalla SQL-lauseketta siten, että ehdosta muodostuu tosi, riippumatta ohjelmakoodissa olevan käskyn rakenteesta. Mikäli koodia ei ole suojattu asianmukaisesti SQL-injektiolta, onnistuu hyökkäys yksinkertaisella lauseenjatkamisella, jossa lauseen ensimmäinen osa voi olla epätosi, mutta loppuosa muodostaa lausekkeeseen tosi ehdon. (Snyder, Myer & Southwell 2010, 33–35.)

Seuraavaksi tarkastellaan muutamia SQL-injektioita hieman perusteellisemmin. Tässä on tarkoituksena konkretisoida hyökkäyksen toiminta kokeellisesti. Tietokannan taulut on toteutettu pelkistetysti, mutta riittävästi SQL-injektion simulointia varten.

Yksinkertainen SQL-hyökkäys

Ensin yritetään selvittää yksinkertaisella kyselyllä, onko hyökkäys mahdollinen. Tietokannassa on testitaulu, jossa on tietokentät id, nimike ja maara (kuva 6).



KUVA 6. Tuotetaulu

WWW-palvelussa on hakutoiminto, jolla voidaan hakea tuotetta nimikkeen mukaan (kuva 7). Hyökkääjä muuttaa kyselyn tarkoituksen kirjoittamalla hakukenttään OR-ehdon, joka muuttaa lausekkeen todeksi riippumatta siitä, onko kyseistä tuotetta taulussa. Mahdollisten vahinko-osumien eliminoimiseksi hyökkääjä voi käyttää hakusanaa, joka ei vastaa mitään tuotetta. Koska lause on tosi, niin WWW-palvelu listaa kaikki taulussa olevat tuotteet.

Hyökkäys ei sinänsä vaikuta kovinkaan vaaralliselta, mutta on huomioitava, että listauksen onnistuessa hyökkääjä voi päätellä, että syötteitä ei ole suodatettu. Päättelyn tuloksena hyökkääjä voi jatkaa toimintaansa vakavammilla hyökkäyksillä.

KUVA 7. Yksinkertainen hyökkäys

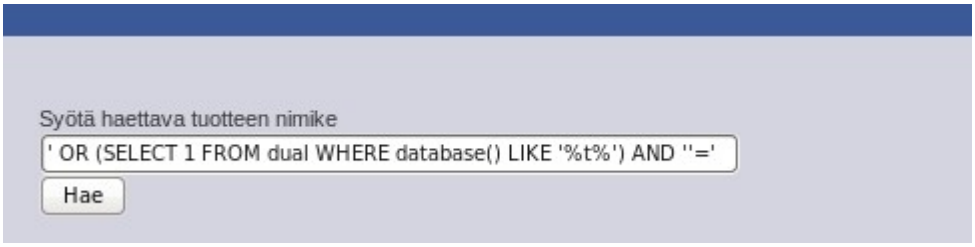
Kuvan 7 esimerkkilause toimii siten, että sanan *eka* eteen ei tule heittomerkkiä, koska kyseinen merkki on jo olemassa lauseen vastaanottavan muuttujan edessä. Sanan *eka* jälkeen tuleva heittomerkki päättää ensimmäisen ehdon, jonka jälkeen muodostetaan toinen ehto, joka on aina tosi. Viimeisen *tosi* sanan jälkeen jätetään heittomerkki pois, koska lauseen vastaanottavan muuttujan takana on jo olemassa ehdon päättävä merkki. (Snyder ym. 2010, 33–36.) PHP-skripti muodostaa esimerkkikuvan hausta seuraavanlaisen SQL-lauseen:

```
SELECT id, nimike, maara FROM testi where nimike like 'eka' OR 'tosi' = 'tosi';
```

Tietokannan nimen selvittäminen

Mikäli ensimmäisessä hyökkäyksessä tulos oli positiivinen, niin jatketaan hyökkäystä ja yritetään selvittää tietokannan nimi. Selvittäminen aloitetaan siten, että arvuutellaan ni-

messä olevia kirjaimia, kunnes kaikki on löytynyt. Etsinnän jälkeen yritetään päätellä löydetyistä kirjaimista tietokannan nimi (kuva 8).



KUVA 8. Tietokannan nimen selvittäminen

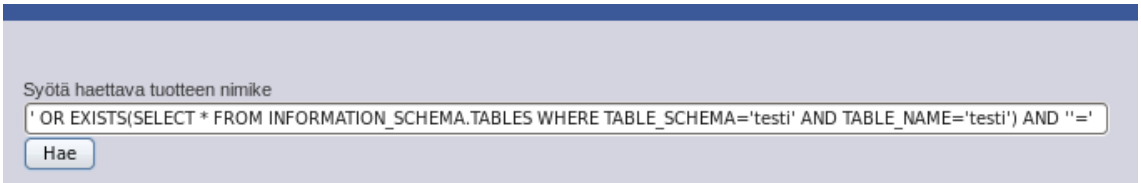
Kuvan 8 esimerkissä kysely kohdistetaan kuvan 6 tauluun. Kyselyssä muodostetaan OR-ehdolla alikysely, jolla haetaan tuotteet taulusta, joka löytyy tietokannasta, jonka nimessä esiintyy kirjain *t*. Mikäli tietokannan nimessä esiintyy *t*, listataan tuloksena kaikki tuotteet. PHP-skripti muodostaa kyselystä seuraavan lauseen:

```
SELECT id, nimike, maara FROM testi where nimike like '' OR (SELECT 1 FROM dual WHERE database() LIKE '%t%') AND ''='';
```

OR-ehtolauseen ensimmäinen osa *(SELECT 1 FROM dual,* hakee osumaa taulusta jossa *dual* vastaa mitä tahansa taulua. Ehtolausekkeen database-funktiolla haetaan tietokanta, jonka nimessä on *t*. Viimeisellä AND-ehdolla lisättiin heittomerkki, jotta lausekkeen syntaksi on oiken PHP:ssä. (SQLzoo.net 2011.)

Taulun nimen selvittäminen

Taulun nimen selvittäminen vaatii enemmän kärsivällisyyttä, koska jokerimerkkejä ei voi käyttää taulua nimeä haettaessa. Koska tietokannan nimi on selvitetty ja WWW-sovelluksen käyttötarkoitus on tiedossa, niin taulun nimen arvaaminen saattaa onnistua melko helposti. Laitetaan kuvan 9 mukainen kysely hakukenttään. Jos nimi on oikein, kysely listaa kaikki tietueet, ja taulun nimi on selvillä. Mikäli tulos on negatiivinen, jatketaan etsimistä uudella sanalla (SQLzoo.net 2011.)



KUVA 9. Taulun nimen selvittäminen

Esimerkissä OR-ehdon haku kohdistetaan virtuaalitietokantaan information_schema, jossa on tiedot kaikista tietokannoista. Virtuaalitietokantaan voidaan kohdistaa kyselyjä, mutta tietokannan tietoja ei voi muokata tai poistaa. Kyselyssä haetaan testi nimisestä tietokannasta testi nimistä taulua. PHP-skripti muodostaa kyselystä seuraavan lauseen:

```
SELECT id, nimike, maara FROM testi where nimike like "OR EXISTS(SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA='testi' AND TABLE_NAME='testi') AND "=";
```

Lauseessa EXISTS palauttaa toden, jos alikyselyn ehto on tosi ja tietueet listataan. Alikyselyn ensimmäinen osa *(SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA='testi'*, kohdistaa etsinnän testi nimiseen tietokantaan ja loppuosa *AND TABLE_NAME='testi')*, etsii testi nimistä taulua. (SQLzoo.net 2011.)

Luvaton kirjautuminen palveluun

Monissa WWW-palveluissa vaaditaan käyttäjältä kirjautumista. Kirjautumistaulussa käytetään yleensä tiivistefunktiota salasanoissa, jotta salasanan lukeminen vaikeutuu tilanteessa, jossa hyökkääjä pääsee käyttäjä tauluun käsiksi. Tämän johdosta kuvassa 7 esitetyn kaltainen yksinkertaisen tosi ehdon laittaminen salasanakenttään ei tuota tulosta.

Tiivistefunktion käyttö parantaa WWW-sovelluksen turvallisuutta, mutta mikäli sovellusta ei ole riittävästi suojattu SQL-injektion varalta, on sekin helppo murtaa. Seuraava esimerkki havainnollistaa, kuinka helposti suojaamattomaan sovellukseen voidaan kirjautua oikeudettomasti, vaikka salasana kentässä käytetään salausfunktiota.

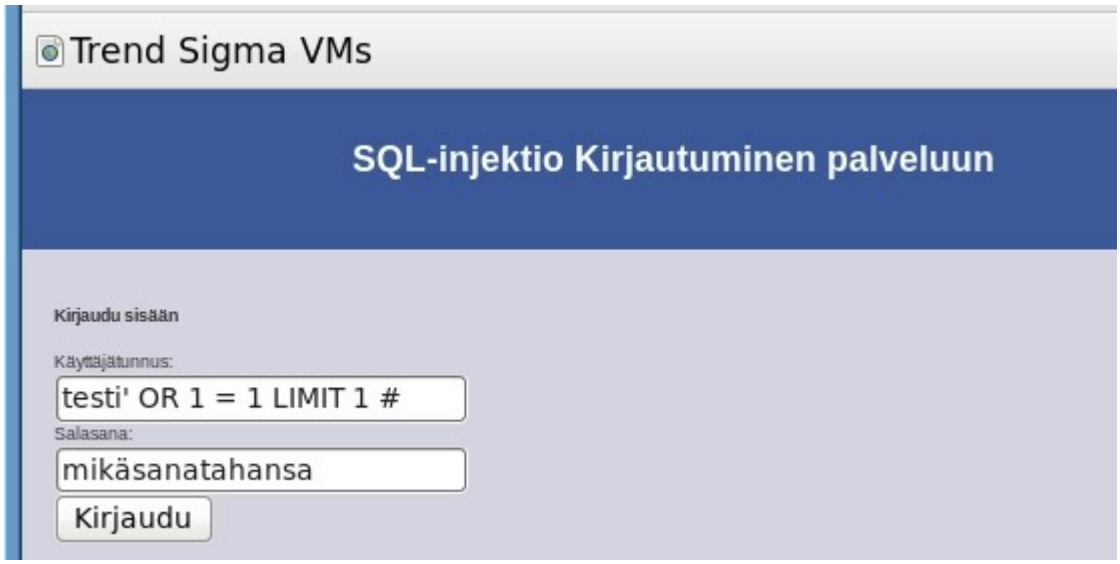
Esimerkissä käytetty salasana on suojattu MySQL-password-funktiolla, joka tekee annetusta salasanasta 41 merkkiä pitkän tiivisteen. Salasanakenttä on jätetty kryptaamatta havainnollisuuden vuoksi.

Tietokannassa on käyttäjille tarkoitettu taulu joka koostuu kentistä id, tunnus ja salasana (kuva 10).

```
mysql> describe kayttaja;
+----------+------------------+------+-----+---------+----------------+
| Field    | Type             | Null | Key | Default | Extra          |
+----------+------------------+------+-----+---------+----------------+
| id       | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| tunnus   | char(30)         | NO   |     | NULL    |                |
| salasana | char(64)         | NO   |     | NULL    |                |
+----------+------------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

KUVA 10. Käyttäjätaulu

Tässä hyökkäyksessä tunkeilija haluaa kirjautua WWW-palveluun oikeudettomasti. Palvelu on suojattu käyttäjätunnuksella ja salasanalla (kuva 11).

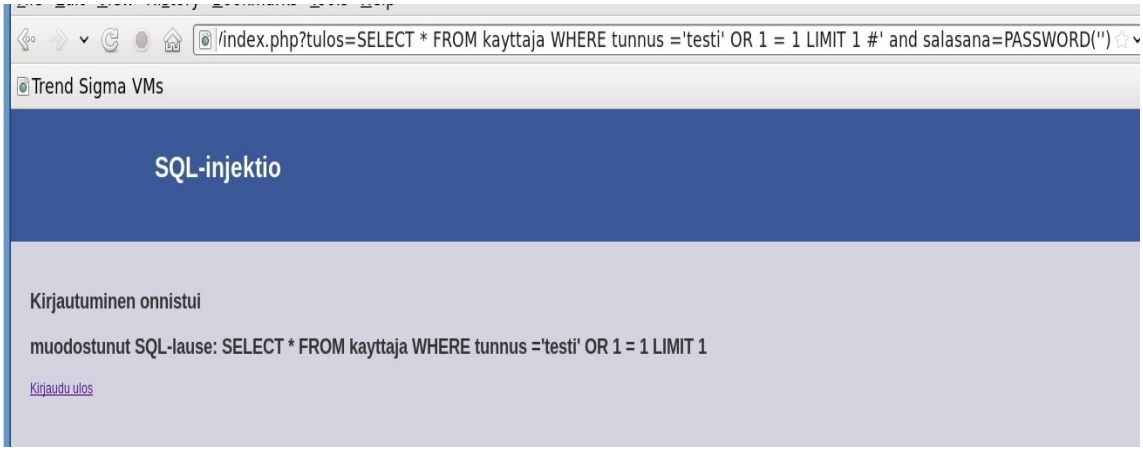


KUVA 11. Luvaton kirjautuminen

Kuvassa 11 testi-sanan jälkeinen lainausmerkki sulkee ensimmäisen ehdon, jonka jälkeen muodostetaan OR-lauseella ehdosta tosi. Komento LIMIT 1 rajoittaa hakutulokset yhteen kappaleeseen. Merkki # muuttaa SQL-lauseen loppuosan kommentiksi, jolloin

salasanan kenttään voidaan kirjoittaa mitä tahansa, koska käytännössä käsky päättyy komennon LIMIT 1 jälkeen. Salasanan kenttä ei tarvitse toiminnan kannalta mitään syötettä, mutta käytännössä kirjautumislomakkeissa vaaditaan molempiin kenttiin arvot.

Kuvassa 12 näkyy osoiterivillä käskyn muoto kokonaisuudessaan lähetettäessä kysely koodin käsittelevälle PHP-sivulle. Kirjautumisen onnistuessa tulostetaan SQL-lause WWW-sivulle.



KUVA 12. Luvattoman kirjautumisen koodit

Järjestelmästä riippumatta on aina äärimmäisen vaarallista, jos ulkopuolinen taho pääsee kirjautumaan oikeudetta järjestelmään. Hyökkäys, jossa järjestelmään tunkeutuja pyrkii hyödyntämään saamiaan tietoja järjestelmällisesti ja huomaamatta mahdollisimman pitkään, on vakavampaa kuin akuutti tuho järjestelmän tietokannassa, joka on asianmukaisesti varmuuskopioitu.

Pääkäyttäjän luominen tietokantaan

Onnistuessaan vaarallisimpia SQL-injektioita on hyökkäys, jossa hyökkääjä luo root-tasoisen käyttäjä tietokantaan. PHP:n oletuksena mysql_query-funktio sallii ainoastaan yhden MySQL-komennon, joten uhka on MySQL-ympäristössä lähinnä teoreettinen. (Snyder ym. 2010, 36–37.) Root-käyttäjällä on kaikki oikeudet tietokantaa eli käyttäjä voi halutessaan suorittaa minkä tahansa operaation tietokannassa (kuva 13). Esimerkissä on käytetty kuvan 1 taulua.

KUVA 13. Oikeudettoman pääkäyttäjän luominen tietokantaan

Kuvan 13 lauseessa yhdistetään hakukenttään kaksi SQL-lausetta. Periaate on sama kuin aikaisemmissakin, eli muodostetaan ehtolause, joka on aina tosi. Ehtolause päätetään puolipisteellä, jonka jälkeen kirjoitetaan lause, joka lisää tietokantapalvelimelle uuden käyttäjän kaikilla oikeuksilla. PHP-skripti muodostaa kyselystä seuraavan lauseen:

```
SELECT id, nimike, maara FROM testi where nimike like '' OR 'tosi'='tosi'; GRANT ALL ON *.* TO 'jasu@%' IDENTIFIED BY 'joku'
```

Ehtolausetta seuraavan lauseen osa *GRANT ALL ON *.* TO 'jasu@%'*, lisää jasu-nimiselle käyttäjälle täydet oikeudet tietokantapalvelimen jokaiseen tietokantaan kaikista IP-osoitteista. Merkit **@%** tarkoitavat kaikkia IP-osoitteita. Lauseen osa *IDENTIFIED BY ' joku'*, määritetään käyttäjälle jasu salasana joku. Hyökkäys ei toimi käytännössä, mutta havainnollistaa hyvin hyökkäyksen yksinkertaisen rakenteen.(Snyder ym. 2010, 37.)

Edelliset esimerkit osoittavat, kuinka helppoa suojaamattomalle WWW-sivuille on hyökätä käyttämällä yksinkertaista koodia syöttökentissä. Ohjemalliset ja automaattiset hyökkäykset ovat huomattavasti tehokkaampia ja vaarallisempia. Vaikka injektiohyökkäykset ovat pitkään tunnettu, niin niitä esiintyy edelleen jatkuvasti.

Viimeisin CERT-FI:n raportoima laaja SQL- injektiohyökkäys, joka ulottui Suomeen, esiintyi 02.12.2011. Hyökkäys saastutti lukuisia tiedostoja eri puolilla maailmaa. Tämän perusteella pääteltiin, että kysymyksessä oli automatisoitu hyökkäys (CERT-FI 2011).

4 HAAVOITTUVUUKSIEN ETSIMINEN JA ESTÄMINEN

4.1 Haavoittuvuuden etsiminen

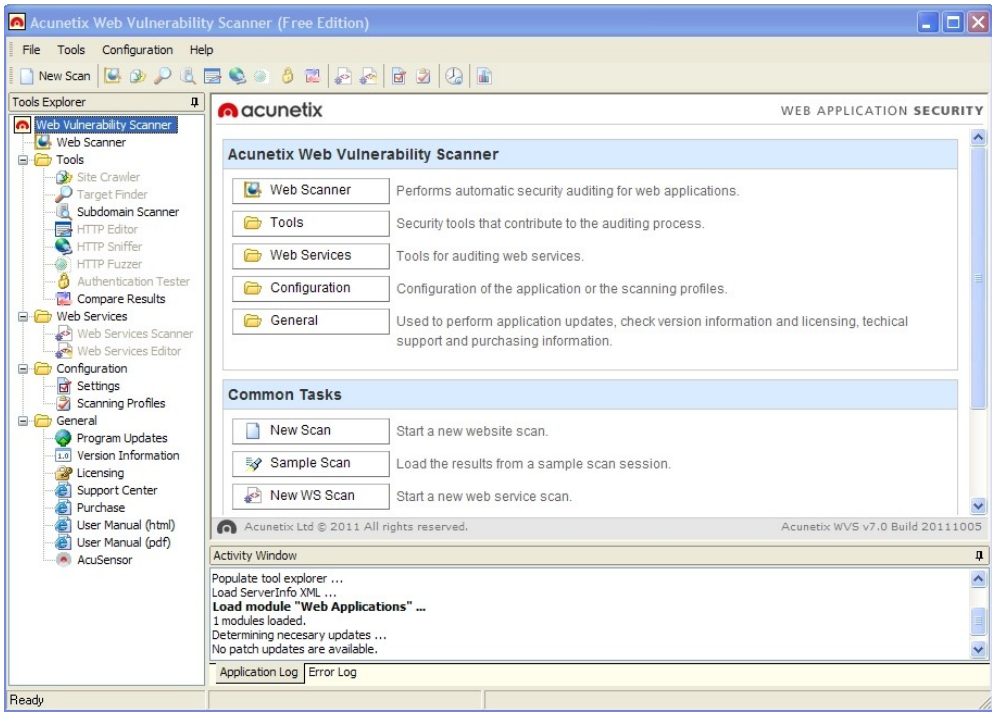
Vaikka haavoittuvuudet pitää huomioida jo sovellusta suunniteltaessa, on sovellus kuitenkin aina testattava haavoittuvuuksien varalta. Edellisiä esimerkkejä käyttämällä voi löytää joitakin haavoittuvuuksia, mutta käytännössä sovellus on testattava asianmukaisesti riippumatta siitä, rakennetaanko uutta sovellusta vai muokataanko vanhaa.

Huolellisen suunnittelun ja toteutuksen lisäksi on syytä käyttää haavoittuvuuksien testaamiseen suunniteltua sovellusta. Sovellus voi olla kokonaisvaltainen, jolla pyritään löytämään kaikki mahdolliset haavoittuvuudet tai sovellus voi olla erikoistunut johonkin tiettyyn osa-alueeseen.

Internetistä löytyy sekä kaupallisia, että ilmaisia sovelluksia, joilla voi testata WWW-sovelluksen haavoittuvuuksia. Käytännössä hyökkäysohjelmatkin voivat olla hyvä mittari etsiessä haavoittuvuuksia, mutta varsin usein niiden mukana yritetään ujuttaa joitain haittaohjelmia, joten en henkilökohtaisesti suosittele niiden käyttöä. Tietoturvaan erikoistuneilla yrityksillä on tarjolla kaupallisia WWW-skannereita, joista on usein saatavana ilmais- tai kokeiluversioita.

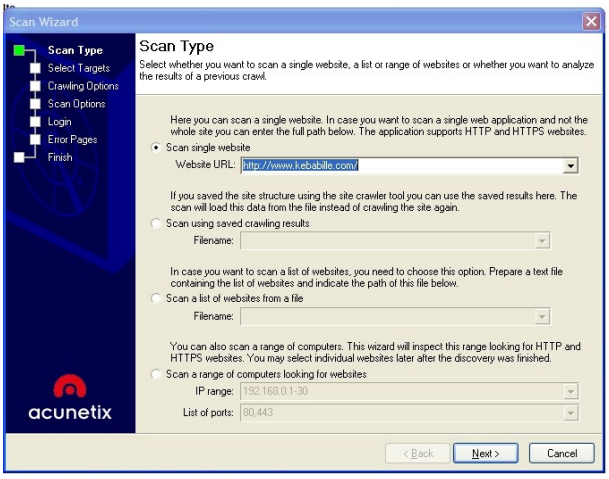
Seuraavassa esimerkissä esitellään Acunetixin maksullisen WWW-skannerin ilmaisversio. Maksullinen ohjelma sisältää paljon toimintoja ja löytää useimmat haavoittuvuudet, mutta ilmaisversiossa toiminnot on rajoitettu ja skannaus etsii ainoastaan XSS-haavoittuvuuksia.

Ohjelmalla skannattiin pikaskannauksella ja perusasetuksilla, suomalainen kohtuullisen suosittu WWW-sivusto. Sivusto on McAfeen sivustoraportin perusteella ongelmaton sivusto. Ohjelma on erittäin helppo käyttää ja ohjelma löytää tiedostot joissa on haavoittuvuuksia. Ohjelman käyttö aloitetaan painamalla työkalurivillä vasemmalla olevaa New Scan -painiketta (kuva 14).



KUVA 14. Acunetixin WWW-skannerin aloitus näkymä

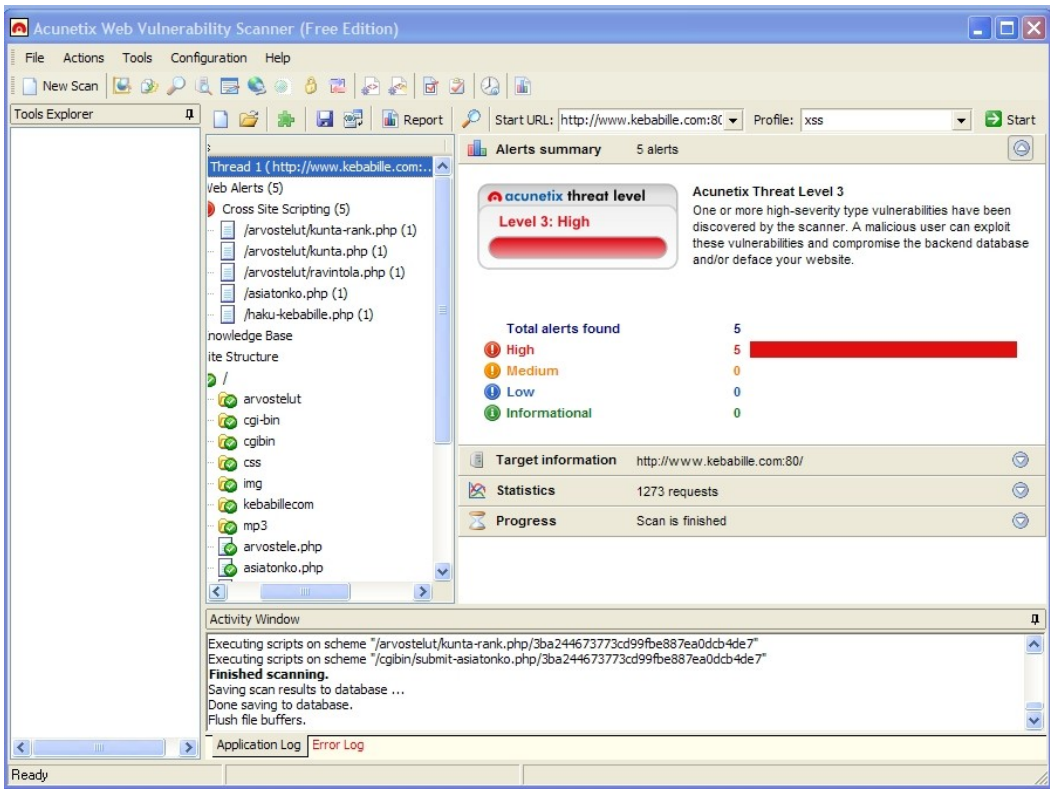
New Scan -painikkeen painamisen jälkeen avautuu skannausvelho, johon asetetaan skannauskohde (Kuva 15).



KUVA 15. Acunetixin WWW-skannerin skannausvelho

Tämän jälkeen jatketaan painamalla Next-painiketta kunnes vastaan tulee Finish-painike. Ensimmäisen Next-painikkeen painamisen jälkeen näkyy yhteenveto skannattavasta sivusta ja seuraavilla Next-painikkeen painalluksilla voidaan muuttaa skannausasetuksia. Lopuksi painetaan Finish-painiketta, jolloin skannaus käynnistyy.

Skannauksen jälkeen avautuu tulosnäkymä, jossa on perusteellinen yhteenveto skannauksen tuloksista (kuva 16).



KUVA 16. Acunetixin WWW-skannerin tulosnäkymä

Kuvassa 16 näkyy oikealla sivustolta löydettyjen uhkien määrä ja vakavuus. Keskellä näkyy tiedostokohtainen yhteenveto, jossa on eritelty haavoittuvuudet sisältävät tiedostot. Huomioitavaa skannauksen tuloksessa on se, että kysymyksessä oli pikaskannaus, joka ei skannaa kaikkia WWW-palvelun osa-alueita. Silti sivustosta löytyi 5 vakavaksi luokiteltavaa haavoittuvuutta.

4.2 Haavoittuvuuden estäminen

Haavoittuvuuksien ehkäiseminen alkaa jo siinä vaiheessa, kun ohjelmoija päättää toteuttaa WWW-sovelluksen. Ensimmäinen pohdittava asia on, halutaanko ylläpitää omaa WWW-palvelinta, vai käytetäänkö webbi-hotellin palveluita. Mikäli käytetään webbi-hotellin palveluita, vastaa palveluntarjoaja palvelimeen liittyvistä turvallisuustekijöistä, kuten ohjelmistoversiot, ohjelmistopäivitykset ym.

Oman palvelimen ylläpidossa kaikki turvallisuusratkaisut jäävät WWW-sovelluksen kehittäjän omalle vastuulle. Palvelimen ylläpitäjän on oltava selvillä käytössä olevan ohjelmistoversion tilasta. Turvallisuuspäivityksiä on seurattava aktiivisesti ja ohjelmistot on päivitettävä heti, kun ohjelmistoon on julkaistu turvallisuus- tai korjauspäivitys. Käytettäväksi ohjelmistoversioksi on syytä valita viimeisin vakaa versio, koska useimmat tietoturva ja muut aikaisemmissa versioissa esiintyneet ongelmat ovat korjattu.

Mikäli jostakin syystä on perusteltua käyttää vanhempia versioita, löytyy www.php.net-sivustolta ja Wikipediasta tiedot vanhemmistakin versioista. Wikipediassa versiot on taulukoitu siten, että niistä saa nopean yleiskuvan PHP:n vanhoista versioista. Taulukoihin on laitettu lyhyt kuvaus version erosta edelliseen ja tieto, onko versio tuettu.

PHP/MySQL-asennusta ei tässä oppaassa käydä läpi luukuunottamatta luvun 4 PHP osiota, jossa käsitellään muutamia tietoturvaan liittyviä peruskonfigurointeja. Kattavia oppaita asennuksesta löytyy sivustolla www.php.net. Sivustolta löytyy myös viimeisin vakaa PHP-versio, viimeisin julkaistu PHP-versio, sekä kattava manuaali PHP-ohjelmoinnista.

Haavoittuvuuksien estämisessä on ensisijaisen tärkeää ymmärtää, miten hyökkääjät toimivat, ja miten hyökkäykset toteutetaan teknisesti. Hyökkäyksien tekniikan ymmärtäminen antaa valmiudet parantaa WWW-sovelluksien tietoturvaa varsin yksinkertaisilla menetelmillä jo sovelluksen rakentamisen alkuvaiheessa. Keskeisin menetelmä hyökkäyksien torjumiseksi on kaiken syöttötiedon huolellinen suodattaminen.

Lähtökohta sovelluksen turvallisuuden varmistamiseksi on se, ettei koskaan saa luottaa dataan. Pahimmillaan kaikki suodattamattomat tiedot, kuten evästeet, Ajax-pyynnöt tai POST-syötteet voidaan väärentää tai manipuloida. Myös luotetuiden käyttäjien syötteet on suodatettava, jolloin tieto on muodollisesti oikeaa, koska muodollisesti oikea syöttötieto varmistaa sen, ettei virheellisesti muotoiltu tieto aiheuta virheitä esim. WWW-sovelluksen tietokannassa. (MacIntyre ym. 2011, 243.)

5 TURVALLINEN WWW-OHJELOINTI

Tässä luvussa käydään läpi tietoturvallisen WWW-ohjelmoinnin peruselementit. Aluksi käsitellään tiedostoihin ja hakemistoihin liittyviä perusasioita, kuten hakemistorakenne, tiedostojen ja hakemistojen käyttöoikeudet ja niiden suojaus. Tämän jälkeen käydään läpi PHP- ja MySQL-perusteita tietoturvallisuuden näkökulmasta. Tässä tavoitteena on selvittää muutamia perusperiaatteita, joiden avulla luvussa 3 esitettyjä hyökkäyksiä voidaan ennaltaehkäistä.

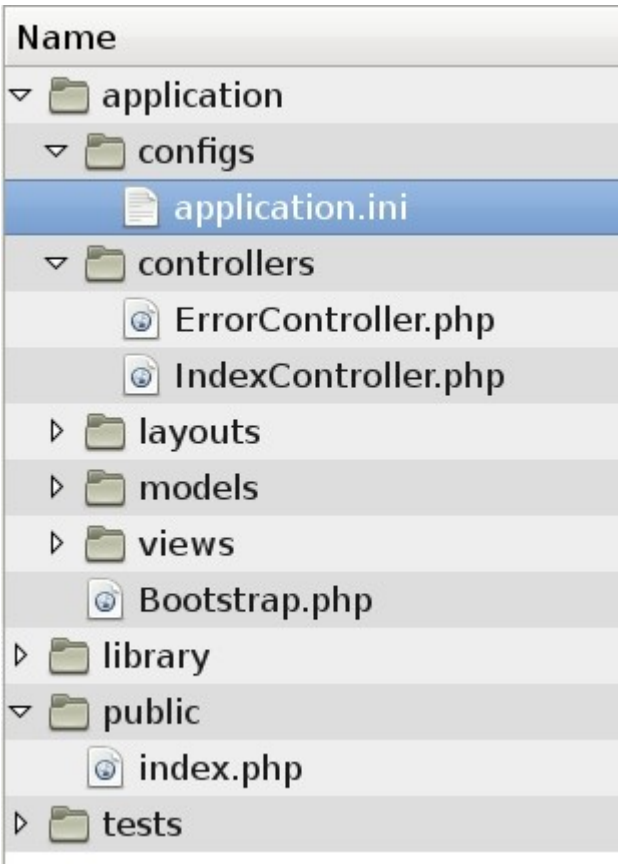
5.1 Tiedostot ja hakemistot

5.1.1 Tietoturvallinen hakemistorakenne

Tietoturvallisen hakemistorakenteen lähtökohta on palvelun luonne, eikä aina voida yksiselitteisesti sanoa, mikä on ainoa oikea ratkaisu. Perusperiaate on kuitenkin, että suunnitteluvaiheessa on oltava selvillä, kenelle tiedostot ja kansiot on tarkoitettu ja minkälaisilla oikeuksilla. Liian suuret rajoitukset voivat aiheuttaa palvelun toimimattomuuden, mutta on kuitenkin varmistettava, ettei kukaan pääse käsiksi oikeudettomasti hakemistoon tai tiedostoon.

Hakemistorakenteessa on aiheellista sijoittaa eri tehtävät eri hakemistoihin, joista yksi on julkinen julkaistavia WWW-dokumentteja varten. Julkiseen hakemistoon on tarkoitus laittaa ainoastaan palvelun käytön kannalta välttämättömät julkiset tiedostot. Muut tiedostot tulee sijoittaa tarkoituksen mukaan omiin hakemistoihin, joihin estetään suora pääsy palvelimella. Tämä parantaa sekä WWW-sovelluksen turvallisuutta sekä helpottaa WWW-sovelluksen hallintaa.

Useat julkaisualustat tarjoavat suhteellisen helppokäyttöisiä ympäristöjä, joissa on pyritty huomioimaan tietoturva. Haavoittuvuuksia kuitenkin löytynee käytännössä jokaisesta julkaisualustasta. Seuraavassa esimerkkinä tarkastellaan Zend Frameworkin hakemistorakennetta, jossa suurin osa tiedostoista on sijoitettu siten, että suora pääsy palvelimella olevaan tiedostoon on estetty (kuva 17).



KUVA 17. zf-työkalun luoma hakemistorakenne

Kuvan 17 hakemistorakenne on luotu zf-työkalulla, joka luo automaattisesti kyseisen hakemistorakenteen. Hakemistot on jaettu neljään päähakemistoon, joista ainoastaan public-hakemistoon on suora pääsy. Tämä parantaa huomattavasti sovelluksen turvallisuutta, koska suora liikenne estetään muihin kansioihin.

Zend Frameworkissa käytetään suunnittelun lähtökohtana MVC-mallia, mutta tässä työssä ei sitä käsitellä. Esimerkin tarkoitus on ainoastaan havainnollistaa turvallista hakemistorirakennetta, jossa pääasia on siinä että, sovelluksen toiminnallisuus, kirjasto- ja testikansioon on estetty suora pääsy.

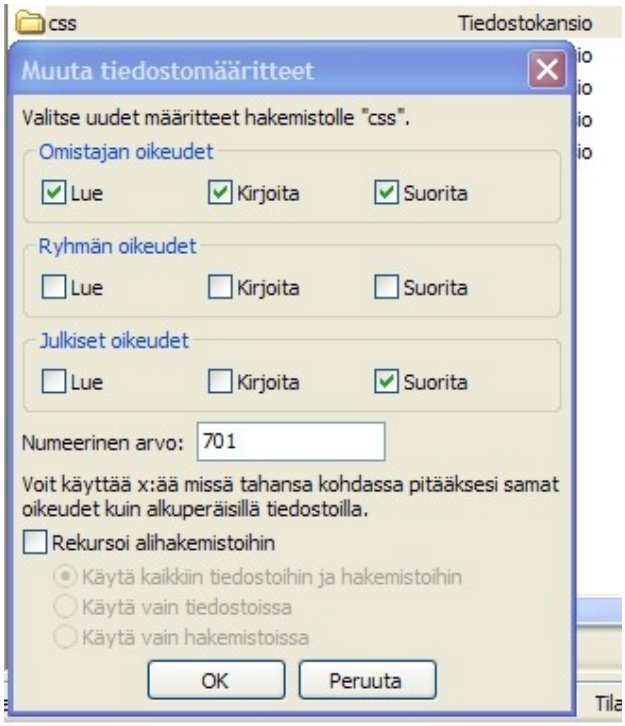
Rakennetta voi soveltaa sovellusta suunniteltaessa, vaikka ei varsinaista Zend Framework -laajennusta käytäisikään. Public-hakemistoon tulee ainoastaan julkiset tiedostot ja tarvittaessa htaccess-asetustiedosto, jolla voidaan ohjata melko monipuolisesti Apachen toimintaa hakemistokohtaisesti esim. suojata tiedostoja ja hakemistoja salasanalla, estää hakemistolistauksia tai uudelleen ohjata käyttäjiä ym.

Tiedostojen ja hakemistojen käyttöoikeudet

Vaikka hakemistorakenne on hyvin suunniteltu, tarvitaan erilaisia käyttöoikeuksia myös julkisessa hakemistossa. Tällaisia tilanteita voivat olla mm. erilaisilla käyttöoikeuksilla varustetut palvelut, joissa on tarkoituksenmukaista rajoittaa hakemistoihin pääsyä eri käyttäjäryhmiltä. Sovelluksessa voi olla tiedostoja eri tarkoituksiin, jotka vaativat erilaisia käyttöoikeuksia, eri tarkoituksiin. Esimerkiksi toiset sivuston käyttäjät voivat lukea tiedoston, mutta kaikki muut toimenpiteet tiedostoon ja hakemistoon on estetty. Toiset käyttäjät voivat lukea ja muokata sivustoa jne.

Käyttöoikeuksien asettaminen saattaa tuntua itsestään selvältä, mutta käytännössä liian suuret rajoitukset saattavat aiheuttaa sovelluksen toimimattomuuden. Käyttöoikeuksia ei kuitenkaan voi antaa perusteettomasti, koska jokainen liian suuri käyttöoikeus heikentää WWW-sovelluksen turvallisuutta. Esimerkiksi WWW-sivustolla on kuvia, jotka ovat kaikkien nähtävissä. Tällainen hakemisto vaatii käyttöoikeudet, jossa käyttäjä ei voi muokata hakemiston sisältöä, mutta voi lukea sitä.

Tiedostojen ja kansioiden käyttöoikeudet on helppo asettaa, tarkastaa ja muuttaa ftp-ohjelmilla, esim. FileZilla, jolloin ylläpitäjän ei tarvitse osata lainkaan Unix-komentoja. FileZillassa käyttöoikeuksien muokkaaminen tarvitsee ainoastaan klikata hiiren oikealla painikkeella käsiteltävää tiedostoa tai kansiota ja valita tiedosto-oikeudet, jolloin avautuu muuta tiedostomääreet -ikkuna. Ikkunasta rastitetaan jokaiseen ryhmään halutut käyttöoikeudet (kuva 18).



KUVA 18 FileZillan ikkuna oikeuksien käsittelyä varten

5.1.2 Tiedostojen ja hakemistojen suojaus (.htaccess)

Tiedostojen ja kansioihin pääsyä voidaan rajoittaa yksinkertaisella HTTP-autentikoinnilla, jolla voidaan suojata hakemistorakenteita tai joustavalla PHP-autentikoinilla, joka on integroitu WWW-sovellukseen (Gilmore 2005, 281–283). Tässä työssä käsitellään pääasiassa HTTP-autentikointia, koska Apachessa menetelmä on helppo ottaa käyttöön. Käyttöönoton helppous saattaa aiheuttaa helposti tilanteen, jossa aloitteleva ohjelmoija käyttää menetelmää ilman riittäviä perusteita.

Apachessa määritellään keskitetysti kaikki palvelimeen liittyvät asetukset mukaanlukien tiedostojen ja kansioiden käyttöoikeuksien hallinta. httpd.conf-tiedostossa, mutta tämän lisäksi asetuksia voidaan antaa hakemistokohtaisesti htaccess-tiedostossa. Httpd.conf-tiedosto ei ole tavallisesti webhotellin asiakkaan muokattavissa, mutta jotkut palvelun-tarjoajat sallivat htaccess-tiedoston käytön tai rajoitetun käytön.

HTTP-autentikointi toimii siten, että käyttäjä yrittää siirtyä suojattuun hakemistoon, jolloin palvelin vastaa 401-koodilla. Selain tunnistaa koodin ja näyttää kirjautumisikkunan,

joka lähettää käyttäjän syötteen takaisin palvelimelle. Käyttäjä saa resurssit käyttöön, mikäli tunnistetiedot ovat oikein. Apachessa HTTP-autentikointi on toteutettu htaccess- ja htpasswd-tiedostoilla. (Gilmore 2005, 281–282.)

.htaccess-tiedosto uudelleenkirjoittaa httpd.conf-tiedostossa olevia asetuksia hakemistokohtaisesti jolloin tiedoston vaikutus on tiedostoton sijaitsevassa hakemistossa ja sen kaikissa alihakemistoissa. .htaccess-tiedoston käyttö on yksinkertaista. Lisätään .htaccess-tiedosto suojattavan hakemistorakenteen juureen ja kirjoitetaan tiedostoon halutut määritykset. Lisäksi tarvitaan .htpasswd-tiedosto, johon luodaan kryptattu salasana htpasswd-ohjelmalla. Esimerkki XAMPP-ohjelman luomasta yksinkertaisesta .htaccess-tiedostosta Windows-ympäristössä (kuva 19).

```
AuthName "xampp user"
AuthType Basic
AuthUserFile "C:\xampp\security\xampp.users"
require valid-user
```

KUVA 19 .htaccess-tiedosto

AuthName määrittää suojatun alueen nimen.
AuthType määrittää HTTP:n tyypin (Basic tai Digest).
AuthUserFile määrittää salasanatiedoston sijainnin.
require valid-user hyväksyy .htpasswd-tiedostossa määritetyt käyttäjät.

.htaccess-tiedostolla voidaan vaikuttaa Apachen toimintaan monilla tavoin, mutta HTTP-autentikointia ei ole tarkoituksenmukaista käyttää esim. WWW-sivujen hallinnoimisessa, koska tällöin aukeaa ylimääräinen pääsy sovelluksen osiin. Hallinta on turvallisempaa tehdä suoraan ftp-yhteydellä tai tarvittaessa tietokantapohjaisella autentikoinnilla.

.htaccess-tiedoston yleisin käyttötarkoitus on rajoittaa joitakin palveluja tietyille käyttäjille tai asettaa hakemistokohtaisia ominaisuuksia tiettyyn osaan WWW-sovellusta. Tyypillinen HTTP-autentikoitu WWW-sivuston osa on esimerkiksi oppilaitoksen materiaalisivut tms.

Turha .htaccess-tiedoston käyttö vaikuttaa negatiivisesti palvelun tehokkuuteen koska useita ylimääräisiä tiedostoja ladataan joka kerta kun sivuostokin ladataan. Lisäksi .htaccess-tiedosto saattaa väärinkäytettynä antaa joillekin käyttäjille mahdollisuuden tehdä palvelinpuolen muutoksia, joita ei aina pystytä kontrolloimaan. Näistä syistä johtuen tiedostoa ei tulisi käyttää perusteettomasti. (Apache Tutorial 2011.)

5.2 PHP

5.2.1 PHP:n peruskonfigurointi

Vaikka palveluntarjoaja suorittaa PHP:n konfiguroinnin, saattaa moni ohjelmoija ylläpitää omaa WWW-palvelinta. Tästä syystä on tärkeää käydä läpi joitakin tietoturvallisuuteen liittyviä perusasioita PHP-konfiguroinnista. PHP:hen sisältyy monipuolinen konfigurointitiedosto, jolla asetuksia voidaan muokata erittäin monipuolisesti, ja ottaa tarvittaessa laajennuksia käyttöön. Lisäksi PHP-jakelu sisältää kaksi erillistä ini-tiedostoa, php.ini-production tuotanto- ja php.ini-development kehitys-ympäristöä varten.

Tiedostot ovat tekstitiedostoja, joiden asetukset ja lisäosat ovat joko poistettu käytöstä muuttamalla asetus kommentiksi lisäämällä puolipiste komennon eteen tai otettu käyttöön jättämällä puolipiste komennon edestä pois. Lähtökohta on, että php.ini-production asetuksia käytetään ainoastaan tuotanto- ja testausympäristössä. Tiedostojen perusasetukset ovat suurelta osin samanlaiset, mutta php.ini.development-asetuksilla sivuston käyttäjä saa paljon tietoa virheistä, joka saattaisi antaa mahdolliselle hyökkääjälle hyödyllistä tietoa tuotantoympäristössä.

Siirrettäessä sivustoa tai WWW-palvelua tuotantokäyttöön, on php.ini-tiedoston ohjeistuken mukaan, aina syytä muuttaa asetukset php.ini-production mallin mukaisesti ja muokata niitä ainoastaan silloin, kun palvelun toiminta tai yhteensopivuus vaatii asetuksiin muutoksia. Taulukossa 1 näkyy php.ini.development- ja php.ini-production-tiedostojen keskeiset erot XAMPP-ympäristössä, jossa käytetään PHP:n versiota 5.3.5.

TAULUKKO 1. php.ini-tiedostojen erot suositusasetuksilla

php.ini-production	**php.ini-development**
error_reporting = E_ALL & ~E_DEPRECATED	error_reporting = E_ALL | E_STRICT
display_errors = Off	display_errors = On
display_startup_errors = Off	display_startup_errors = On
track_errors = Off	track_errors = On
html_errors = Off	html_errors = On
mysqlnd.collect_memory_statistics = Off	mysqlnd.collect_memory_statistics = On
session.bug_compat_42 = Off	session.bug_compat_42 = On
session.bug_compat_warn = Off	session.bug_compat_warn = On

Taulukon 1 tiedostojen vertailussa selviää, että julkaisupalvelimen asetuksista ovat virheilmoitukset asetettu pois-tilaan, kun vastaavasti kehityspalvelimella näytetään kaikki virheilmoitukset ja varoitukset. Seuraavaksi lyhyt kuvaus em. asetuksista.

Error_reporting-asetuksella määritetään PHP:n raportointitarkkuuden taso, joita on 16 (Gilmore 2005, 28–30). Tasoja voidaan yhdistää tarvittaessa boolean-operaattorilla. Taulukon 1 esimerkissä tiedostossa php.ini-production on error_reporting-asetus asetettu siten, että raportointi koskee ajonaikaisia virheitä, mutta poistaa kaikki virheilmoitukset, jotka koskevat varoituksia toiminnoista tai ominaisuuksista, jotka raportoivat, että ominaisuus on vanhentunut. Tiedostossa php.ini-development vastaava asetus on asetettu näyttämään kaikki virheet. Molemmissa tiedostoissa käytetään boolean-operaattoreita, jolla voidaan tehdä yksityiskohtaisia määrittelyjä tarpeen mukaan yhdistelemällä eri raportointitasoja.

Display_errors-asetus määrittää, sallitaanko error_reporting-asetuksella määritettyjen virheiden tulostaminen (Gilmore 2005, 28–30).

Display_startup_errors-asetus määrittää, sallitaanko PHP:n käynnistysprosesduureille ominaisten virheiden näyttäminen (Gilmore 2005, 28–30).

Track_errors-asetus määrittää, tallennetaanko viimeinen virhesanoma $php_error_msg-muuttujaan (Gilmore 2005, 28–30).

Html_errors-asetus määrittää, sallitaanko virhesanomien sulkeminen HTML-tageilla. (Gilmore 2005, 28–30).

Mysqlnd.collect_memory_statistics-asetus määrittää, sallitaanko MySQL-muistinkäytön tilastointi (PHP Manual 2012a).

Session.bug_compat_42-asetus määrittää, alustetaanko session muuttujat globaaleiksi, vaikka register_globals-asetus on pois. Ominaisuus poistettu versiosta 5.4.0. (PHP Manual 2012b.)

session.bug_compat_warn-asetus määrittää, varoitetaanko session.bug_compat_42-ominaisuudesta. Toiminto vaatii *session.bug_compat_42*-ominaisuuden olevan päällä. (PHP Manual 2012b.)

Virheilmoituksien lisäksi on käytävä läpi muita turvallisuutta parantavia asetuksia. Register_globals-asetus saattaa olla merkittävä tietoturva uhka, varsinkin alustamattomana (MacIntyre ym. 2011, 259). Asetuksella voidaan alustaa lomaketiedot globaaleiksi muuttujiksi. Asetus on oletuksena pois päältä PHP:n verisosta 4.2.0 lähtien, mutta silti on syytä varmistaa asetuksen tila. Ominaisuus on poistettu versiosta 5.4.0 (PHP Manual 2012c). Suositeltava asetus tuotantoympäristössä on *register_globals = Off.*

Magic_quotes_gpc-asetuksella määritetään käytetäänkö syötteissä automaattisia karkausmerkkejä. Toiminto saattaa aiheuttaa tietokannassa epäjohdonmukaisuuksia, josta johtuen asetus on syytä olla pois päältä ja suorittaa toiminto tietokanta funktioilla (MacIntyre ym. 2012, 259). Periaatteessa toiminto parantaa tietoturvaa, mutta saattaa aiheuttaa tietokannassa ongelmia. Suositeltava asetus tuotantoympäristössä on *Magic_quotes_gpc = Off.* Ominaisuus on vanhentunut ja on poistumassa tulevista PHP-versioista.

Session.use_trans_sid on oletuksena pois päältä, koska sen käyttö vaarantaa tietoturvaa. Mikäli asetus laitetaan päälle, niin istuntotunnus voidaan asettaa keksiin tai osoiteriville

(MacIntyre ym. 2012, 259). Suositeltava asetus tuotantoympäristössä on *session.use_trans_sid = 0.*

Disable_Clasess-funktio liittyy olio-ohjelmointiin, jota ei tässä käsitellä. Funktiolla voidaan tarvittaessa estää haluttujen luokkien käyttö (MacIntyre ym. 2012, 259). Esimerkki luokan käytön estämisestä: *disable_clasess = ”joku”* (poistetaan joku niminen luokka käytöstä).

Disable_functions-asetuksella poistetaan kaikki vaaralliset funktiot, jotka eivät ole välttämättömiä sovelluksen toiminnan kannalta (MacIntyre ym. 2012, 259). Esimerkki funktion poistamisesta: *disable_functions = curl_exec.*

Kaikkia PHP-direktiivejä tässä ei käyty läpi, koska ne muuttuvat jatkuvasti. Vanhoja direktiivejä poistetaan ja uusia tuodaan tilalle. Palvelinasetuksia konfiguroidessa on syytä tarkistaa käytössä olevan PHP-version php.ini-tiedosto ja selvittää, mitkä php.ini-production php.ini-development tiedostojen keskeiset erot, ja mitkä ovat konfiguraation suositusasetukset.

Tämän lisäksi kaikki tarpeettomat direktiivit ovat mielestäni hyvä poistaa käytöstä. Esimerkkinä voidaan mainita file uploads -direktiivi, joka on turha sellaisessa tapauksessa, jolloin tiedostoja ei siirretä selaimen kautta palvelimelle. (MacIntyre ym. 2012, 259.)

5.2.2 PHP-ohjelmointi

Vaikka luvussa 4 käytiin läpi pääperiaatteita haavoittuvuuksien löytämisestä ja ehkäisystä, on syytä tarkastella yksinkertaisia menetelmiä, joilla voidaan parantaa tietoturvaa koodintasolla. Tietokantaluokkia (PDO) ja olio-ohjelmointia ei tässä oppaassa käsitellä, koska työ on rajattu käsittelemään perusteita.

Syötteen validointi

Syötteiden tarkastus on ensimmäinen ja ehkä tärkein yksittäinen toimenpide, jolla ohjelmoija pystyy parantamaan sovelluksen tietoturvaa. Validointi voidaan tehdä erilaisilla funktioilla, joista puhutaan lisää SQL-injektion torjumisen yhteydessä.

Syötteen validointia helpottaa PHP-versiosta 5.3.2 lähtien oleva filter_var-funktio, jolla voi validoida helposti syötteitä. Esimerkiksi sähköpostin tai url-osoitteen validointiin ei enää tarvitse käyttää säännöllisiä lausekkeita, koska validointi on helppo tehdä em. funktiolla. Seuraavassa esimerkissä selviää, kuinka helppoa URL-osoitteen suodattaminen on filter_validate-funktiolla (PHP Manual 2012d).

```
<?php
   $url=”http://www.tamk.fi”;
   if(filter_var($url, FILTER_VALIDATE_URL)){
         print ”OK”;
    }else{
         print ”Virheelinen URL-osoite”;
    }
?>
```

KOODIESIMERKKI 1. Funktion filter_var käyttö

Sivun lataus

Sivun lataukseen on kiinnitettävä huomiota, jos siirrytään jollekin sivulle ja ladataan siihen sisältöä toisesta tiedostosta esim. $_GET-metodilla.

Seuraavissa esimerkeissä tarkastellaan, miten osoiterivin parametrina annettu $_GET-syöte voidaan suojata. Mikäli suojaus ei ole asianmukainen, pääsee hyökkääjä halutessaan käskisi kaikkiin tiedostoihin, joihin käyttöoikeudet riittävät. Muutetaan ainoastaan parametrina annettu $_GET- arvo selaimen osoiterivillä. Periaatteessa myös WWW-sivut voidaan sisällyttää sivustoon, mutta testiympäristössä WWW-osoitteen syöttäminen parametriksi selaimen osoiteriville ei toiminut.

Testi.php-tiedostoon välitetään $_GET-metodilla PHP-tiedosto eka.php, joka on tarkoitus sisällyttää testi.php-tiedostoon (koodiesimerkki 2).

```
http://localhost/testit/testi.php?tunnus=eka.php          (alkuperäinen osoite)
http://localhost/testit/testi.php?tunnus=../testi.txt     (hyökkäys osoite)

<?php
  // puutteellisesti toteutettu sivun sisällyttäminen
  if(isset($_GET['tunnus'])){
        include($_GET['tunnus']); // sisälyttää $_GET-muuttujalla annettu sivu
  }
?>
```

KOODIESIMERKKI 2. Tiedoston sisällyttäminen

Koodissa otetaan $_GET-muuttujan sisältö vastaan sisältöä suodattamatta tai tarkastamatta sitä riittävästi. Lisäksi osoiterivillä näkyy suoraan, että tunnuskentän arvo on tiedosto eka.php. Hyökkääjän tarvitsee ainoastaan kirjoittaa osoiteriville eka.php-arvon tilalle *../testi.txt*. Mikäli tiedosto- ja kansio-oikeudet eivät ole kunnossa, niin tuloksena ladataan selaimelle WWW-kansion yläkansiosta testi.txt-tiedoston sisältö.

Edellinen hyökkäys on erittäin vaarallinen, koska hyökkääjällä on mahdollisuus lukea kaikki suojaamattomat tiedostot. Hyökkäys on kuitenkin varsin helppo estää. Asianmukaiset tiedosto-oikeudet sekä syötteen käsittely, ehkäisee tehokkaasti em. hyökkäyksen. Tässä tapauksessa tarvitsee lisätä ainoastaan yksi IF-ELSE-lause, jolla voidaan estää tiedostojen oikeudeton tutkiminen (koodiesimerkki 3).

Lisäksi vältetään antamasta suoraa osoitetta osoiteriville, eli annetaan $_GET-muuttujan arvoksi, jokin arvo. Tässä käytetään sanaa *eka.* Testi.php-tiedosto ottaa arvon vastaan ja vertaa arvoa toiseen. Mikäli arvot vastaavat toisiaan, sisällytetään eka.php-tiedosto testi.php-tiedostoon. Mikäli arvot eivät vastaa toisiaan niin lopetetaan suoritus.

```
http://localhost/testit/testi.php?tunnus=eka              (alkuperäinen osoite)
http://localhost/testit/testi.php?tunnus=../testi.txt              (hyökkäys osoite)

if(isset($_GET['tunnus'])){
        if($_GET['tunnus']==="eka"){  // tutkitaan vastaanotettu arvo
          include(eka.php);                  // jos tosi, niin ladataan eka.php
        }else{
          exit("Virhe");
        }
  }
```

KOODIESIMERKKI 3. Syötteen tutkiminen

XSS-hyökkäyksien estäminen

XSS-hyökkäyksen estäminen onnistuu suodattamalla syöttötiedot htmlspecialchars-funktiolla, joka muuttaa HTML-merkit entiteeteiksi (MacIntyre ym. 2011, 251). Esimerkki htmlspecialchars-funktion käytöstä: *htmlspecialchars($muuttuja, ENT_QUOTES, 'UTF-8').*

Joissakin tapauksissa saattaa olla tarpeen käsitellä yksittäisiä merkkejä tai merkkijonoja. Toiminto onnistuu käyttämällä str_replace-funktiota (PHP Manual 2012e). Esimerkiksi jossakin tapauksessa saattaa olla tarpeen tuottaa jotakin sisältöä sivustolle, mutta joitakin vahingollisia merkkejä on karsittava. Esimerkki str_rplace-funkition käytöstä: *$muuttuja=str_replace($kommentti, ”<”, ”<”);*. Suodatus on hyvä tehdä siten, että suodatetaan tiedot mahdollisimman tiukasti tarkoituksensa mukaan, jolloin ainoastaan turvallinen ja muodollisesti oikea syöttötieto päästetään läpi.

Istunnon kaappaus voidaan ehkäistä tehokkaasti generoimalla istunto uudelleen. Koodiesimerkissä 4 generoidaan session-tunnus uudelleen, säilyttäen kuitenkin istuntotiedot (MacIntyre ym. 2011, 252).

```
<?php
session_start();  // aloitetaan istunto
session_regenerate_id();  // korvataan id uudella id:llä
```

KOODIESIMERKKI 4. Session uudelleen generointi (MacIntyre ym. 2011, 252)

CSFR-hyökkäyksen estäminen

CSFR-hyökkäyksen voi estää luomalla salaisen id-tunnuksen ja sisällytämällä se piilotettuun lomakekenttään (koodiesimerkki 5). Koodiesimerkissä luodaan csrf_token istuntotunnus. Istuntotunnuksessa käytetään SHA1-tiivistefunktiota ja uniqid-funktiota, joka perustuu kellonaikaan sekä satunnaislukuarvoon. Muodostettu arvo sijoitetaan piilotetun lomakkeen *value*-kenttään. Koodiesimerkissä 6 verrataan lomakkeen *value*-kentän ja csrf_token arvoja. Mikäli arvot vastaavat toisiaan varmistetaan, että toiminto on suoritettu tietyssä ajassa. (MacIntyre ym. 2011, 252.)

```
<?php

session_start();
session_regenerate_id();
if ( !isset( $_SESSION['csrf_token'] ) ) {
  $csrf_token = sha1( uniqid( rand(), true ) );  // luodaan salainen istuntotunnus
  $_SESSION['csrf_token'] = $csrf_token;  // sijoitetaan session taulukkoon
  $_SESSION['csrf_token_time'] = time();  // asetetaan aika session taulukkoon
}
?>
<form>
<input type="hidden" name="csrf_token" value="<?php echo $csrf_token; ?>" />
</form>
```

KOODIESIMERKKI 5. Luodaan salainen istuntotunnus (MacIntyre ym. 2011, 252)

```
<?php

session_start();
if ( $_POST['csrf_token'] == $_SESSION['csrf_token'] ) {  // verrataan
  $csrf_token_age = time() - $_SESSION['csrf_token_time'];  // lasketaan aika

  if ( $csrf_token_age <= 180 ) { //three minutes //
      //valid, process request
  }
}
?>
```

KOODIESIMERKKI 6. Verrataan lomakkeen arvoa session arvoon (MacIntyre ym. 2011, 252)

SQL-injektio

Kuten XSS-hyökkäyksen torjunnassa, myös SQL-injektion torjunnassa on olennaista syöttötiedon huolellinen käsittely. Tietokantaan tehtävissä kyselyissä on varmistettava, että tieto on muodollisesti oikeaa, eikä vahingollisia merkkejä pääse tietokantaan. Esimerkiksi syötettäessä tietokantaa luku, on varmistettava, että se on numeerinen tietotyyppi. Lisäksi on varmistettava, että syöttötiedon pituus on oikea. (Snyder ym. 2010, 38.)

PHP sisältää funktioita, joilla syötteen tarkastus on helppo tehdä (taulukko 2). Mikäli on tarkoituksenmukaista käyttää tarkempaa suodatusta, voidaan käyttää esim. säännöllisiä lausekkeita, joilla voidaan suodattaa syöttötieto yksityiskohtaisesti. Taulukossa 2 on esimerkkejä joistakin tietotyypin tutkivista funktioista (PHP Manual 2012f).

TAULUKKO 2. Tietotyypin tutkiva funktio

Tyypin tutkiva funktio	**Toiminta**
is_string();	Palauttaa toden jos syöte on merkkijono
is_float()	Palauttaa toden jos syöte on liukuluku
is_int()	Palauttaa toden jos syöte on kokonaisluku
is_bool()	Palauttaa toden jos syöte on boolean
is_object()	Palauttaa toden jos syöte on objekti

Koodiesimerkissä 7 tutkitaan, onko syöte kokonaisluku. Mikäli syöte ei ole kokonaisluku, niin keskeytetään suoritus. (Snyder ym. 2011, 22). Taulukossa 2 olevat funktiot toimivat koodiesimerkissä 7 olevan syntaksin mukaisesti.

```
$numero = $_POST['kokonaisluku'];

if ( !is_int( $numero ) ) exit ( "$numero ei ole kokonaisluku" );
```

KOODIESIMERKKI 7. Tutkii onko syöte kokonaisluku

Koodinvaihto on tehokas tapa estää MySQL-injektio. Koodinvaihto voidaan suorittaa funktiolla mysql_real_escape_string(). Funktio lisää kenoviivat erikoismerkkien eteen, lukuun ottamatta prosentti- ja alaviivamerkkejä. Mikäli Magic_quotes_gpc-asetus on päällä, eikä sitä voi muuttaa, niin automaattisesti asetetut kenoviivat voidaan poistaa stripslashes-funktiolla. Tämän jälkeen mysql_real_escape_string() toimii odotetulla tavalla.

Koodiesimerkissä 8 käydään läpi, miten hyökkäysyritys voidaan torjua, vaikka salasana kentässä ei olisi käytetty tiivistefunktiota. Koodi toimii riippumatta siitä, onko get_magic_quotes_gpc päällä vai ei.

```
<?php
$tunnus = "fff ' or 1 = 1 LIMIT 1 #"; // hyökkäysyritys
$ssana = joku;
If(get_magic_quotes_gpc()){   // jos get_magic_quotes_gpc() on toiminnassa
        $tunnus=stripslashes($tunnus);          // niin poistetaan kenoviivat
        $ssana=stripslashes($ssana);
}
// suoritetaan kysely
$tunnus=mysql_real_escape_string($tunnus); // laitetaan kenoviivat
$ssana=mysql_real_escape_string($ssana);
// suoritetaan kysely
$sql="SELECT * FROM kayttaja WHERE tunnus ='$tunnus' and ssana='$ssana'";
$result=mysql_query($sql);
```

KOODIESIMERKKI 8. Syötteen suodatus

Koodiesimerkki 4 tuottaa seuraavan tuloksen, jossa muuttujan $tunnus OR-ehdon edessä oleva vahingollinen ′-merkki eliminoidaan käyttämällä kenoviivaa. Tuloksena muodostuu seuraavanlainen SQL-lause: *SELECT * FROM kayttaja WHERE tunnus ='fff\'or 1 = 1 LIMIT 1 #' and salasana='joku'*. Kenoviiva aiheuttaa sen että vahingollinen *or*-ehto poistuu kyselystä.

Edellinen esimerkki ei riitä turvaamaan tietoturvaa, koska salasanan tulee olla aina kryptattu jollakin tiivistefunktiolla. Tiivistefunktion ensisijainen tarkoitus on, ettei tietokannassa olevia tietoja pystytä helposti väärinkäyttämään. Lisäksi tiivistefunktio parantaa sovelluksen tietoturvaa esim. tilanteessa, jossa koodivaihto on saattanut jäädä vahingossa suorittamatta.

Kirjautumissivulla jossa käytetään tiivistefunktiota sekä käyttäjätunnus- että salasanakentässä (koodiesimerkki 5), ei koodiesimerkissä 4 esitetyn kaltainen hyökkäys onnistu. Tiivistefunktio tekee kaikista merkkijonon sisältävistä merkistä tiivisteen, vahingolliset merkit mukaan lukien. Koodiesimerkissä 5 muutetaan edellistä kyselyä siten, että se käyttää SHA1-tiivistefunktioita sekä tunnus- että salasanakentässä.

```
// suoritetaan kysely
$sql="SELECT * FROM kayttaja WHERE tunnus =sha1('$tunnus') and salasana=sha1('$ssana')";
```

KOODIESIMIRKKI 5. Syötteensuodatus ja tiivistefunktio

Koodiesimerkissä 5 hyökkäys ei pääse läpi, koska tietokantaan syötetyn kyselyn muuttujien arvo muutuu 40 merkkiä pitkäksi tiivisteeksi. Kysely tuloksena muodostuu seuraava SQL-lause: *SELECT * FROM kayttaja WHERE tunnus =sha1('fff or 1 = 1 LIMIT 1 #') and salasana=sha1('salasana').*

Kysely etsii tietokannasta seuraavia arvoja:

tunnus: 74b799e702a34e45af0fef265613ce82919490dc

salasana: 899fa932e51efda70aeaffbab6d3a712cfbaaf6e

Kuvassa 20 näkyy erilaisten tiivistefunktioiden muodostamia tiivisteitä, kun testissä on käytetty *testi* sanaa.

KUVA 20. Tiivistefunktioita

Tiivistefunktion käyttö on tarpeen esim. salasanojen, ja muiden arkaluonteisten tietojen suojaamiseksi, mutta niiden turhaa käyttöä on vältettävä. Lisäksi on päätettävä, mitä tiivistefunktiota käytetään. Kuvassa 20 ilmenee, että MD5-tiivistefunktio kuluttaa huomattavasti vähemmän resursseja, kuin SHA-2(512). MD5-tiivistefunktio on kuitenkin varisin turvaton, eikä sitä pitäisi käyttää sellaisenaan.

PHP:ssä voidaan käyttää melko yksinkertaista menetelmää, joka parantaa tiivistefunktioiden turvallisuutta huomattavasti. Menetelmässä käytetään kellonaikaan perustuvaa yksilöllistä tunnusta ja satunnaislukua. Ensin luodaan yksilöllinen tunnus johonkin muuttujaan, jonka jälkeen se liitetään jo tiivistetyn salasanan kanssa tiivistefunktiolla. Yksilöllinen tunnus tallennetaan tietokantaan. Kirjauduttaessa palveluun yksilöllinen

tunnus liitetään salasanatiivisteen kanssa, jolloin oikea salasana löytyy (MacIntyre ym. 2010, 260).

```
// salasanan vahvistaminen suolaamalla
<?php
  $salt = uniqid( mt_rand() ); // haetaan satunnaislukuarvo
  $password = md5( $user_input );  // tiivistetään käyttäjän antama salasana
   $stronger_password = md5( $password.$salt );  // yhdistetään salasan ja suola ja tii
                                                 // vistetään ne
?>
```

Koodiesimerkki 6. Salasanan vahvistaminen (MacIntyre ym. 2010, 260)

Vaikka edelläesitetyt esimerkit eivät ole monimutkaisia, parantaa niiden soveltaminen omaan ohjelmointiin merkittävästi WWW-sovelluksen tietoturvaa varsin vaatimattomalla työmäärällä.

6 MYSQL

Tässä luvussa käydään läpi joitakin perusasioita MySQL-tietokannan turvallisuuden parantamiseksi. SQL-injektioita tässä ei enään käsitellä, koska ne ovat käsitelty aikaisemmissa luvuissa. Pääasia on MySQL-tietokannan käyttöoikeuksissa.

MySQL-tietokannan turvaaminen perustuu pitkälti eritasoisiin käyttöoikeuksiin. Käyttöoikeuksia ei saa koskaan antaa enempää, kuin palvelun käyttö edellyttää. Liian suuret käyttöoikeudet aiheuttavat aina potentiaalisen tietoturvariskin. Vaikka PHP:ssä pyritään ehkäisemään tietokantaan kohdistuvat hyökkäykset, ei se riitä vielä täysin turvaamaan MySQL-tietokantaa.

6.1 MySQL:n saantiteidot

MySQL-saantitiedot perustuvat tietokantaan tallennettuihin tauluihin, joista keskeiset autentikoinnin ja oikeuksien tarkastamisen kannalta ovat seuraavat (taulukko 3) (Gilmore 2005, 594).

TAULUKKO 3. Oikeuksien saantitiedot (Gilmore 2005, 594)

user	Määrittää kirjautumiseen oikeutetut käyttäjät ja isäntäkoneet
db	Määrittää tietokantojen käyttöön oikeutetut käyttäjät
host	db-taulun laajennos, joka täydentää isäntäkonenimet
tables_priv	Määrittää taulukohtaiset oikeudet.
columns_priv	Määrittää sarakekohtaiset oikeudet

MySQL-palvelimen asennuksessa yhteydessä luodaan tietokannalle root-käyttäjä, jolla on täydet oikeudet tietokantaan. Vanhemmissa MySQL-versioissa root-käyttäjän salasana jätetään oletuksena tyhjäksi (Gilmore 2005, 591), mutta uudemmissa versioissa (tes-

tattu versiossa 5.1.41-3ubuntu12.10) asennusohjelmassa on käyttöliittymä, jossa ehdotetaan salasanan antamista root-käyttäjälle.

Ensimmäiseksi on varmistettava, että root käyttäjällä on riittävän vahva salasana. Mikäli salasanaa ei ole annettu asennuksen yhteydessä, on se annettava välittömästi MySQL-palvelimen ensimmäisen käynnistämisen jälkeen. Salasanoihin pätevät samat suositukset kuin muissakin salasanasuosituksissa. Salasanan tulee olla riittävän pitkä sekä sisältää numeroita ja erikoismerkkejä. Lisäksi salasana ei tule olla merkityksellinen sana.

Hyvä esimerkki vahvasta salasanasta on se, että siirretään sormet näppäimistöllä esim. yksi näppäin oikealle ja kirjoitetaan sana kymmensormijärjestelmällä, jolloin esim. *jasu* nimestä tulee *ksdi* (MySQL Documentation 2011a). Esitetyllä menetelmällä kirjoitettu riittävän pitkä sana, jossa on käytetty numeroita ja erikoismerkkejä, vaikeuttaa salasanahyökkäystä merkittävästi.

Kirjautuessa tietokantaan root-käyttäjänä, voidaan helposti todeta, onko root-käyttäjälle määritetty salasanaa. Kirjautumien suoritetaan komennolla *mysql -u root -p*. Mikäli salasanaa ei ole määritetty, niin kirjautuminen onnistuu root-oikeuksin komennolla *mysql -u root* kaikilta käyttäjiltä jotka haluavat kirjautua tietokantapalvelimelle. Jos salasana on määritetty, niin kirjautumisen yhteydessä kysytään salasanaa (MySQL Documentation 2012a). Salasana määritetään komennolla *SET PASSWORD FOR root@localhost = PASSWORD('salasana');*. Komennossa *root* on (pää)käyttäjä ja *@localhost* on palvelin.

Salasanan antaminen komentoriviltä ei ole aivan ongelmatonta, koska joku saattaa nähdä sen. Siksi on tärkeää varmistaa, ettei kukaan näe annettua salasanaa tai pääse katsomaan komentoja salasanojen antamisen jälkeen. Työskentelyn jälkeen on aina muistettava kirjautua ulos MySQL-ympäristöstä *exit*-komennolla. Jos poistutaan hetkeksikään koneelta, on tietokone. Lukittava. Lisäksi voidaan varmistaa, ettei kukaan pääse katsomaan terminaalin historiatietoja tyhjentämällä terminaalin historia komennolla *history -c*.

Root-käyttäjän salasanan lisäksi on varmistettava, ettei mikään asiaton taho pääse käsiksi MySQL-saantitietoihin, kuten user-tauluun. Jos asiaton taho pääsee käsiksi esim. user-tauluu, aiheutuu kriittinen tietoturvauhka tietokantaan, koska em. taulussa säilyte-

tään käyttäjätilit, joissa on käyttäjätietoja esim. käyttäjien salasanoja. (MySQL Documentation 2012a).

6.2 MySQL:n oikeuksien hallinta

GRANT- ja REVOKE-komennolla root-käyttäjä voi hallita käyttäjien ja ryhmien oikeuksia. GRANT-komennolla myönnetään ja REVOKE-kommennolla poistetaan oikeuksia. Yleensä luodaan ensin käyttäjä tietokantaan, jonka jälkeen määritetään käyttäjälle oikeudet (MySQL Documentation 2012b).

Käyttäjä luodaan tietokantaan ilman käyttöoikeuksia komennolla *CREATE USER 'joku'@'localhost' IDENTIFIED BY 'salasana'*. Komennon osa *IDETIFIED BY 'salasana'* määrittää salasanan käyttäjälle. Vaikka salasanan voi määrittää käyttäjille myöhemmin, on salasanan määrittäminen kuitenkin syytä tehdä käyttäjän luonnin yhteydessä. Kun käyttäjä on luotu, määritetään käyttäjälle käyttöoikeuksia tarpeen mukaan.

Mikäli käyttäjien luominen ilman oikeuksia ei ole perusteltua, on mielestäni on parempi tapa luoda käyttöoikeudet käyttäjän luonnin yhteydessä komennolla *GRANT select, insert ON testi.kayttaja TO 'joku'@'localhost' IDENTIFIED BY 'salasana';*. Komento luo *joku*-nimisen käyttäjän, jolla on oikeudet hakea ja lisätä tietoja ainoastaan *testi*-tietokannan *kayttaja*-tauluun. Jos käyttäjä on luotu etukäteen, edellisestä lauseesta jätetään osa *IDENTIFIED BY 'salasana'* pois.

Määritettäessä käyttäjäoikeuksia, on muistettava, että oikeuksia on erilaisia. Lisäksi erilaisia oikeustasoja on aina tietokantaan yhdistämisestä saraketasoon asti. Tästä johtuen on kiinnitettävä erityistä huomiota, ettei anneta vahingossa esim. tietokantatason oikeuksia taulutason oikeuksien sijaan.

Jos käyttäjälle on annettu liikaa oikeuksia, on niitä vähennettävä. Oikeuksia vähennetään *REVOKE*-komennolla, joka toimii samaan tapaan kuin GRANT, mutta päinvastoin. Komennolla *REVOKE select, insert ON testi.kayttaja FROM 'joku'@'localhost'* poistetaan tietojen haku- ja lisäysoikeudet käyttäjältä *joku* tietokannan testi kayttaja-tauluun.

Tarkoituksenmukaisen käyttöoikeuksien toteuttamisen lisäksi tietoturvaa voidaan parantaa erilaisilla resurssirajoituksilla. Resurssirajoituksia määritetään lähteen MySQL Documentation (2012c) mukaan seuravilla komennoilla:

max_connections_per_hour määrittää, montako yhteyttä voidaan enintään ottaa tunnissa.

max_questions_per_hour määrittää, montako select-kyselyä voidaan enintään tehdä tunnissa.

max_updates_per_hour määrittää, montako insert- ja update-päivitystä voidaan enintään tehdä tunnissa.

max_user_connections määrittää, montako samanaikaista yhteyttä voidaan enintään ottaa.

Resurssirajoitukset määritetään GRANT-komennolla muiden oikeuksien määrittelyn yhteydessä käyttämällä WITH-komentoa. Komento *GRANT select, insert ON testi.kayttaja TO 'joku'@'localhost' WITH MAX_UPDATE_PER_HOUR 5;* määrittää käyttäjälle *jasu* tietojen haku- ja lisäysoikeudet tietokannan *testi* tauluun *kayttaja*. Lisäksi päivityskerrat (insert, update) ovat rajoitettu viiteen. (Gilmore 2005, 610.)

Olennainen toiminto oikeuksien hallinnoinnissa on oikeuksien tutkiminen, joka onnistuu valitsemalla MySQL-tietokanta ja listaamalla tables_priv-taulun sisältö. Lisäksi voidaan käyttää komentoa *SHOW GRANTS FOR*, jolla voidaan tutkia yksittäisen käyttäjän oikeuksia (Gilmore 2005, 608). Komento *SHOW GRANTS FOR joku@localhost;* näyttää käyttäjän *joku* oikeudet ja resurssirajoitukset. Käyttäjät voivat käyttää em. komentoa tarkastellessaan omia oikeuksia, mutta ainoastaan pääkäyttäjällä tulisi olla oikeudet käyttää MySQL-tietokantaa.

6.3 MySQL:n varmuuskopiointi

Tietoturvasta puhuttaessa on aina muistettava varmuuskopiointi, jolla varmistetaan tiedon eheys. MySQL sisältää apuohjelman, jolla voidaan kopioida tietokanta helposti. Ensin siirrytään *mysql/bin* hakemistoon, jonka jälkeen kirjoitetaan mysqldump-komento:

mysqldump --user=root --password testi> testi.sql tai
mysqldump -u root -p testi> testi.sql (kopioi testi-tietokannan testi.sql-nimiseen tiedostoon)

mysqldump --user=root --pasword testi testi> testi.sql (kopioi testit-tietokannan testi nimisen taulun testi.sql-tiedostoon.

Kopioidut SQL-tiedostot ovat tekstitiedostoja, joita voi muokata tarvittaessa kuten muitakin tekstitiedostoja. Varmuuskopioidun tietokannan tai taulun voi palauttaa yhtä helposti kun kopioidakin. Palauttamiseen käytetään MySQL-komentoa.

mysql -u root -p testi < testi.sql (palauttaa testitietokantaan testi.sql nimisen varmuuskopion)

7 MUUTA TIETOTURVAAN LIITTYVÄÄ

Luvussa 3 esitellyt hyökkäykset on hyvä muistaa myös tavanomaisessa Internet-surffauksessa. Merkillepantavaa on se, että monet hyökkäykset onnistuvat huolimattomasta Internet-käyttäytymisestä. Näkemykseni mukaan vastuu ei ole pelkästään koodaajan, vaan myös käyttäjien on tiedostettava, mitkä uhat verkossa uhkaavat.

Tietoturvan ymmärtäminen päivittäisessä Internet-käyttäytymisessä parantaa tietoturvaa kokonaisuudessa, koska tietoturvan kokonaisvaltainen ymmärtäminen auttaa huomioimaan tietoturvan aloitettaessa esim. omien WWW-sivujen suunnittelua tai aloitettaessa opiskelemaan WWW-suunnittelua tai -ohjelmointia.

Hyökkäysmenetelmät ovat varsin yksinkertaisia ja niiden torjuminen ei ole tekniikasta kiinni. Voidaan perustellusti kysyä, miten esimerkiksi XSS-hyökkäys, joka on esiintynyt julkisuudessa jo vuonna 2000, voi vieläkin toimia. Vastaus lienee, että monet itseoppineen ohjelmoijat ovat aloittaneet tuottamaan WWW-palveluja tai siirtyneet työskentelemään alan yrityksissä. Toimintatapojen muuttaminen jälkikäteen ei ole itsestään selvä asia, josta johtuen tietoturva käsitteellisellä tasolla tulisi ymmärtää jo Internetin perukäytön yhteydessä.

Tietoturvaan liittyvistä uhkista sekä onnistuneista hyökkäyksistä raportoidaan varsin näkyvästi ja niihin on suhtauduttava vakavasti. Sekä peruskäyttäjä että WWW-suunnittelija voi omalta osaltaan parantaa tietoturvaa verkossa lukemalla tietoturvaan liittyviä julkaisuja ja seuraamalla aktiivisesti esim CERT-FI-sivuston tiedotteita verkkohyökkäyksistä ja -uhkista.

8 POHDINTA

Toteutettu opas noudatti osittain alkuperäistä näkemystä, josta uskon olevan hyötyä useimmille lukijoille. Tavoitteena oli tuottaa tietoturvan perusopas, jossa näkökulma oli tarkastella tietoturvaa aloittelevan PHP/MySQL-ohjelmoijan ja ohjelmoinnista ja tietoturvasta kiinnostuneiden näkökulmasta. Oppaan tarkoitus oli pyrkiä yleispätevyyteen, jolloin oppaassa esitettyjä ratkaisuja voidaan soveltaa joiltakin osin muihinkin ohjelmointiympäristöihin. Lisäksi oppaan tarkoitus oli auttaa kohderyhmää tiedostamaan tietoturvan perusteita ja auttaa ymmärtämään tietoturvan merkitys WWW-ohjelmoinnissa.

Oppaassa lähestyttiin käsiteltävää ongelmaa siten, että pyrkimys oli tuoda WWW-hyökkäys mahdollisimman lähelle lukijaa. Tästä johtuen esimerkkejä oli runsaasti ja mielestäni riittävästi. Esimerkit olivat pelkistettyjä ja ne olivat mahdollisuuksien mukaan testattu testiympäristössä. Testiympäristö oli rakennettu tietoisesti turvattomaksi, jotta testihyökkäykset onnistuivat ja niistä saatiin kuvankaappauksia, joilla hyökkäysmenetelmää pyrittiin korostamaan. Oman näkemykseni mukaan visuaalinen ilmaisu tuottaa tehokkaamman tuloksen, jolloin lukijan on helpompi sisäistää viestitettävä asia, kuin pelkillä koodiesimerkeillä tekstin seassa.

Työtä tehdessäni annoin työni luettavaksi muutamalle tutulle peruskäyttäjälle, jotka yllättyivät, kuinka helppoa on yksinkertaisen verkkohyökkäyksen toteuttaminen. Palautteen perusteella opinnäytetyössä käytetyt kuvat olivat avainasemassa hyökkäyksien ymmärtämisen kannalta. Tästä johtuen uskon, että valitsemani lähestymistapa käsiteltävään ongelmaan oli onnistunut.

Opasta työstäessäni suurin ongelmani oli työn rajaus, koska opas oli rajattu liian väljästi. Tilanteesta johtuen, oli hankaluutenani päättää, mitkä asiat kuuluvat perusoppaaseen ja mitkä soveltuvat ohjelmointiin enemmän perehtyneille. Ongelma pelkistyy siihen, että oppaasta ei tullut ehkä tarpeeksi tiivistä, eli puutteita oppaasta varmasti löytyy.

Ennen opinnäytetyön tekemistä en ole perehtynyt kovinkaan paljoa käsiteltyyn aiheeseen ja opin työstä paljon uutta, mitä en ole varsinaisesti ajatellut aikaisemmin. Haavoittuvuuksien yleisyys yllätti, vaikka kysymyksessä oli melko vanhat ja tunnetut hyökkäysmenetelmät. Testasin Acunetixin web-skannerilla noin 25 satunnaisesti valittua

WWW-sivua, joista 5:ssä ilmoitettiin olevan ainakin yksi vakava XSS-haavoittuvuus. Tulos vakuutti minut opinnäytetyön hyödyllisyydestä, mutta samalla herää kysymys, mitenkä tulos on mahdollinen. Vastauksia voi olla esim. päivittämätön julkaisualusta tai kiireesti toteutettu WWW-palvelu. Joka tapauksessa tulos osoittaa, ettei tietoturvaa ole sisäistetty käsitteellisellä tasolla riittävästi.

Yhteenvetona voin todeta, että opinnäytetyö työ vastaa suurelta osin tarkoitustaan, vaikka lopputulos ei olekaan täydellinen. Työ auttaa lukijaa tiedostamaan tietoturvan merkityksen sekä antaa työkaluja turvallisempaan WWW-ohjelmointiin. Tuoreet verkkolähteet sekä runsaat testit antavat tuloksille uskottavuutta,

LÄHTEET

Apache Tutorial 2012. [WWW-sivu]. Luettu 15.1.2012.
http://httpd.apache.org/docs/current/howto/htaccess.html

Calin, B. 6.10.2009. [WWW-sivu]. Luettu 10.1.2012.
http://www.acunetix.com/blog/news/statistics-from-10000-leaked-hotmail-passwords

CERT-FI 2011. [WWW-sivu]. Luettu 8.11.2011.
http://www.cert.fi/tietoturvanyt/2011.html

Facebook 2012. [WWW-sivu]. Luettu 16.3.2012.
http://newsroom.fb.com/content

Gilmor, J. 2005. PHP & MySQL – Tehokas hallinta. Suom. Kuvaja, A., alkuperäinen teos 2005. Helsinki: Readmi.fi.

Järvinen, P. 2002. Tietoturva & yksityisyys. 2. painos. Jyväskylä: Docendo.

Lehto, T. 2010.[WWW-sivu]. Luettu 28.12.2011
http://www.tietokone.fi/uutiset

MacIntyre, P., Danchilla, B. & Gogala 2011. Pro PHP Programming. New York City: Apress.

Microsoft 2011. [WWW-sivu]. Luettu 26.01.2011.
http://office.microsoft.com/fi-fi/outlook-help

MySQL Documentation 2012a. [WWW-sivu]. Luettu 10 3.2012.
http://dev.mysql.com/doc/refman/5.0/en/security-guidelines.html

MySQL Documentation 2012b. [WWW-sivu]. Luettu 10.3.2012.
http://dev.mysql.com/doc/refman/5.6/en/grant.html

MySQL Documentation 2012c. [WWW-sivu]. Luettu 10.3.2012.
http://dev.mysql.com/doc/refman/5.6/en/user-resources.html

Oikeusministeriö 2008. [WWW-sivu]. Luettu 10.9.2011
http://www.om.fi/Etusivu/Ajankohtaista/Uutiset

Ou, G. 2011. Online services security report card. Luettu 10.9.2011.
http://www.digitalsociety.org/2010/11

OWASP 2011a. [WWW-sivu]. Luettu 6.11.2011.
https://www.owasp.org/index.php/XSS

OWASP 2011b. 2011. [WWW-sivu]. Luettu 6.11.2011.
https://www.owasp.org/index.php/DOM_Based_XSS

PHP Manual 2012a. [WWW-sivu]. Luettu 15.2.2012
http://php.net/manual/en/mysqlnd.config.php

PHP Manual 2012b. [WWW-sivu]. Luettu 15.2.2012
http://php.net/manual/en/session.configuration.php

PHP Manual 2012c. [WWW-sivu]. Luettu 15.2.2012
http://php.net/manual/en/security.globals.php

PHP Manual 2012d. [WWW-sivu]. Luettu 9.3.2012
http://php.net/manual/en/filter.filters.validate.php

PHP Manual 2012e. [WWW-sivu]. Luettu 9.3.2012
http://php.net/manual/en/function.str-replace.php

PHP Manual 2012f. [WWW-sivu]. Luettu 10.3.2012
http://www.php.net/manual/en/ref.var.php

Siu, E. 2007. [WWW-sivu]. Luettu 10.1.2012.
http://blogs.msdn.com/b/esiu/archive/2007/09/22/http-header-injection-vulnerabilities.aspx

Snyder, C., Myer, M. & Southwell. 2010. Pro PHP Security. From Application Security Principles to the Implementation of XSS Defenses. New York City: Apress.

SQLzoo.net 2011. [WWW-sivu]. Luettu 6.11.2011.
http://sqlzoo.net/hack/24table.htm

Vaalisto, H. 2011. [WWW-sivu]. Luettu 20.3.2012.
http://www.digitoday.fi/data

Viestintävirasto, 2011. [WWW-sivu]. Luettu 26.10.2011.
http://www.ficora.fi/index/palvelut/palvelutaiheittain/tietoturva.html

Wikipedia 2011a. [WWW-sivu]. Luettu 9.9.2011
http://en.wikipedia.org/wiki/Twitter

Wikipedia 2011b. [WWW-sivu]. Cross site scripting. Luettu 28.12.2011
http://fi.wikipedia.org/wiki/Cross_site_scripting

Wikipedia 2012. [WWW-sivu]. Facebook. Luettu 16.3.2012
http://en.wikipedia.org/wiki/Facebook